Oracle Rdb7

データベース・パフォーマンスおよびチューニングのためのガイド

リリース 7.0

2001 年 12 月

部品番号 : J03806-01

ORACLE®

Oracle Rdb7 データベース・パフォーマンスおよびチューニングのためのガイド , リリース 7.0

部品番号 : J03806-01

原本名：Oracle Rdb7 Guide to Database Performance and Tuning Release7.0

原本部品番号：A41747-1

目次

2 データベース・パフォーマンス分析ツール

3 パフォーマンスの要因の分析

4 パラメータの調整

6 VMScluster 環境で Oracle Rdb を使用する方法

7 チューニングの概念と方法論

8 データベース・リソースのボトルネックの診断

A Oracle Rdb の論理名と構成パラメータ

はじめに

Oracle Rdb は、リレーショナル・データ・モデルに基づく汎用のデータベース管理システムです。

このマニュアルの目的

このマニュアルでは、データベース分析の方法論について解説し、パフォーマンスの問題を特定、分析、分離、解決するためのアプローチを順を追って説明します。データベース・パフォーマンスを左右する要因、その要因をデータベース分析ツールを使用して調べる方法、データベース・パラメータを調整してパフォーマンスを改善する方法について説明します。

このマニュアルでは、様々なデータベース・チューニング・ツールおよびユーティリティを使用して、システム、ユーザー、データベース・リソースの統計を報告する方法について説明します。この種のツールには、Oracle Rdb Performance Monitor と Oracle Trace ソフトウェアがあります。Oracle Trace ソフトウェアは、イベントベースのデータを収集し、数種類のレポートを作成します。Oracle Trace を使用すると、Oracle Expert for Rdb ソフトウェア用の作業負荷の入力データも作成できます。

Oracle Expert for Rdb ソフトウェアでは、作業負荷、データ・ボリューム、システム環境の情報を指定すると、いつでも物理的に最適なデータベース設計が生成されます。

対象読者

このマニュアルは、データベース・パフォーマンスの維持や改善の責務を負う経験豊富なデータベース管理者を対象としています。読者は、データ処理手順、データベース管理に関する基本的な概念と用語、OpenVMS オペレーティング・システムに精通しており、Oracle Rdb を熟知していることを前提としています。

このマニュアルの構成

1 章（1-1）	パフォーマンスの要因について説明し、パフォーマンスの分析に使用するユーティリティとツールを紹介し、さらにパフォーマンス分析の方法を示します。
2 章（2-1）	RMU Analyze コマンド、Performance Monitor、Oracle Rdb 論理名、Oracle Trace for OpenVMS ソフトウェア、Oracle Expert for Rdb など、データベース・パフォーマンスの分析に使用するツールの概要を示します。
3 章（3-1）	デフォルトのパラメータ、ディスク I/O、データの分散など、データベース・パフォーマンスに関する一般的な検討事項について説明します。After-image ジャーナル、ロック、インデックス検索がパフォーマンスにどう影響するかについても詳しく説明します。
4 章（4-1）	データベース・パラメータとオペレーティング・システム・パラメータを調整してパフォーマンスを改善する方法を示します。
5 章（5-1）	クエリー・オプティマイザの概要を示し、オプティマイザがデータ検索に使用するアクセス方法を説明します。オプティマイザの効果を調整する方法についても解説します。
6 章（6-1）	VMScluster 環境で Oracle Rdb データベースを構成する方法について説明します。
7 章（7-1）	データベース・チューニングについて説明し、チューニングの対象とタイミングを決定する際に役立つチューニングの方法論を示します。
8 章（8-1）	データベース・リソースのボトルネックの診断に役立つ一連のデシジョン・ツリーを示します。
付録 A（A-1）	Oracle Rdb の論理名と構成パラメータについて詳しく説明します。
付録 B（B-1）	Oracle Trace for OpenVMS ソフトウェアが提供する Oracle Rdb イベント・データ・テーブルについて説明します。
付録 C（C-1）	RDMS$DEBUG_FLAGS 論理名と RDB_DEBUG_FLAGS 構成パラメータを使用して、オプティマイザの検索方法、クエリーの実行、クエリーのコストを調べる方法を説明します。

関連マニュアル

Oracle Rdb の詳細については、このドキュメント・セットを構成する他のマニュアル（特に次のマニュアル）を参照してください。

- Oracle Rdb7 Introduction to SQL
- Oracle Rdb7 SQL プログラミング・ガイド
- Oracle Rdb7 SQL Reference Manual
- Oracle RMU Reference Manual
- Oracle Rdb7 Guide to Database Design and Definition
- Oracle Rdb7 Guide to Database Maintenance
- Oracle Rdb7 Installation and Configuration Guide
- Oracle Rdb7 日本語リリースノート

『Oracle Rdb7 日本語リリースノート』には、Oracle Rdb ドキュメント・セットを構成するすべてのマニュアルの一覧が掲載されています。

次のドキュメント・セットには、関連する情報が記載されています。

- Oracle Expert for Rdb ドキュメント・セット
- The Oracle Trace for OpenVMS ドキュメント・セット
- Oracle CDD/Repository ドキュメント・セット
- オペレーティング・システムのドキュメント・セット

表記規則

例では、特に指定のない限り、各行の終わりには改行があるものとします。各行を入力したら、最後に [Enter] キーを押してください。

例では、多くの場合プロンプトは表示されていません。一般に、インタラクティブ・シーケンスを正確に示す必要がある場合は表示されますが、それ以外の場合は文やコマンド自体を重視するためにプロンプトは省略します。

このマニュアル内の VMScluster 環境に関する記述は、特に指定のない限り、VAX ノードのみで構成する VAXcluster システムと、1 つ以上の Alpha ノードを含む VMScluster システムの両方に適用されます。

このマニュアルで OpenVMS とは、OpenVMS Alpha オペレーティング・システム、OpenVMS VAX オペレーティング・システム、VAX VMS オペレーティング・システムを表します。

このマニュアルでは、オペレーティング・システムまたはプラットフォームに固有の情報を識別するアイコンを使用します。複数のプラットフォームまたはオペレーティング・システムに関連する場合は、アイコンを組み合せるか汎用アイコンを使用します。次に例を示します。

アイコンは、Compaq Tru64 UNIX オペレーティング・システムに固有の情報の始まりを示します。

このアイコンの組合せは、OpenVMS VAX オペレーティング・システムと OpenVMS Alpha オペレーティング・システムに固有の情報の始まりを示します

◆ この菱形の記号は、特定のオペレーティング・システムまたはプラットフォームに固有の情報の終わりを示します。

このマニュアルでは、次の表記規則も使用します。

[Ctrl]+[X]	例でこの記号が使用されている場合は、[Ctrl] キーを押した状態で特定の文字キー（[X]）を押すことを表します。
[Enter]	例でこの記号が使用されている場合は、[Enter] キーを表します。
[Tab]	例でこの記号が使用されている場合は、[Tab] キーを表します。
⋮	例に縦の省略記号がある場合は、例に直接関連しない複数の行が省略されていることを表します。
...	文中やコマンド中の水平の省略記号は、文やコマンド内で例に直接関連しない要素が省略されていることを表します。
e、f、t	印刷されたマニュアルの索引欄には、ページ番号の後に小文字の e、f、t が記載されている場合があります。e、f、t は、それぞれ、例、図、表の参照を表します。
< >	山カッコは、ユーザーが指定する名前を囲みます。
[ ]	大カッコは、オプションの句を囲みます。1 つを選択するか、指定を省略することができます。
$	ドル符号は、OpenVMS では DIGITAL Command Language プロンプトを表し、Compaq Tru64 UNIX では Bourne シェル・プロンプトを表します。

1

データベース・パフォーマンスの概要

データベースを実装し、本番に移行すると、そのデータベースは、毎日使用されることで、データベース設計を論理面と物理面の両方から継続的にテストされることになります。最終的に、ユーザーがデータベースの応答が遅くなったことを報告する場合もあります。たとえば、初めは秒単位だった標準的なレポートの生成時間がやがては分単位になることがあります。パフォーマンスの低下は徐々に発生することも、一夜にして発生することもあります。慢性的または断続的な場合もあります。すべてのアプリケーションやユーザーに影響をおよぼすことも、一部にしか影響しないこともあります。このマニュアルは、データベース・パフォーマンスの問題を診断し、解決するためにご利用ください。

この章では、データベース・パフォーマンスとチューニングの概念を紹介します。パフォーマンスを左右する要因と、問題を分離するためのツールについて簡単に説明します。1.4 項（1-10）には、パフォーマンスの問題の原因を論理的に検出するためのガイドラインとなる、パフォーマンス分析の方法論を示します。

7 章（7-1）と 8 章（8-1）には、6 章以前の各章の補足説明があります。7 章と 8 章では、チューニングの概念を詳細に定義し、システム、データベース、アプリケーションのチューニングによってデータベース・パフォーマンスがどう変わるかを詳しく調べます。8 章（8-1）では、パフォーマンスの問題の特定、分析、分離、解決、および得られたソリューションの監視に役立つ一連のデシジョン・ツリーを示します。オラクル社では、6 章以前の記述を理解した上で 7 章（7-1）と 8 章（8-1）を読むことをお薦めします。

1.1 パフォーマンスとチューニング

データベース・パフォーマンスとは、広義にはユーザーの満足度を表します。実際的なパフォーマンスの期待値を設定するのは管理者の責務です。ユーザーは、応答時間が多くの要因によって変化し得ること、リソースの制限など、管理者が制御できない要因もあることを認識する必要があります。特定のデータベースとシステム構成における最適なパフォーマンスは、最も一般的なデータベース操作に関するできるだけ短い応答時間と定義されます。

このマニュアルでは、データベース・パフォーマンスを左右する要因のうち、管理者が分析し、調整できるものについて説明します。ロック、データ分散、I/O などのデータベース・パフォーマンスの特性を監視し、評価すれば、1.2 項（1-2）で説明する 1 つ以上の要因を調整することでパフォーマンスを改善できるかどうかを判断できます。

データベース・パフォーマンスを左右する要因を調整する作業は、データベースのチューニングと呼ばれます。チューニングとトラブルシューティングは別の作業です。ソフトウェア・エラーによって生成されるバグチェック・ダンプ・ファイルの分析などのトラブルシューティングについては、『Oracle Rdb7 Guide to Database Maintenance』を参照してください。チューニングには、データベースに対するデータの読取りと書込みの効率を左右するパラメータの調整が含まれます。

データベースは、初期設計段階で一度チューニングされますが、実装後に新たなチューニングが必要になる可能性があります。実装後のチューニングは、次の項目の 1 つ以上が発生した場合に行われることもあります。

- ユーザーのクレーム
- データベースの成長
- データベースの物理的な特性を変える変更
- パフォーマンスの監視
- リソースの追加または削減
- 新しいアプリケーションまたは作業負荷の変更
- ユーザーの追加

最初の物理設計時には、データベースを作成する際にパフォーマンス・データを利用できません。データベース作成者は、『Oracle Rdb7 Guide to Database Design and Definition』に示すガイドラインに従うことで、可能な限り適切な物理設計を行うことができます。このマニュアルでは、読者はすでにデータベースを運用しており、既存の設計を改善する必要があると想定しています。チューニングの詳細については、7 章（7-1）を参照してください。

1.2 パフォーマンスの要因

パフォーマンスの低下はよくある問題です。しかし、多くの場合、パフォーマンスの問題の原因は明白ではありません。データベース・パフォーマンスの問題の原因になり得る一般的な領域を、表 1-1（1-3）に示します。

表 1-1 パフォーマンスの問題の領域

問題の領域	調査項目
システム・リソース	メモリー、ディスクの数、CPU の能力はサポートするユーザー数に対して十分か
メモリー管理	パラメータ値はデータベース・アプリケーションにアクセスするユーザー・プロセスに合わせてチューニングされているか
オペレーティング・システム・パラメータ	システム・パラメータ値の設定は適切か
プロセス・パラメータ	ユーザー・アカウント・クォータの設定は適切か
論理的なデータベース設計	特定の行の検索または更新に時間がかかりすぎないか
物理的なデータベース設計	論理データベース設計は効率的に実装されているか ■ パラメータ。サイトに固有のニーズに合わせて Oracle Rdb パラメータを変更したか。 ■ 行の配置。行の断片化やずれが多すぎないか。ページの空き領域が少なすぎないか。ハッシュ・バケットのオーバーフローが多すぎないか。ハッシュ・バケットがあるページとは異なるページに移動したデータ行が多くないか。 ■ データベースの空き領域。データベースの空き領域が少なすぎないか。記憶領域のエクステントが多くないか。初期領域割当てサイズが増大しすぎていないか。 ■ ロックの問題。同時に同じ行の更新または読取りを試行するプログラムが多すぎないか。
アプリケーションの設計	SQL プリコンパイラと SQL モジュール言語プログラムはデータベースを効率的に使用しているか。テーブルは十分考慮して作成されているか。トランザクションは最大の並行性を得るように設計されているか。

問題の領域がわかると、その領域内でデータベース・パフォーマンスを左右する要因の評価を開始できます。この項では、これ以降、問題が発生し得る領域について簡単に説明し、詳細を参照するページを示します。

1.2.1 システム・リソースとメモリー管理

CPU の能力、ストレージ、メモリーのいずれかが不十分な場合は、応答に非常に長い時間がかかることがあります。次の資料を参照し、パフォーマンスの問題がシステム・リソースの不足によるものかどうかを判断してください。

OpenVMS VAX OpenVMS Alpha

- このマニュアルの 1.3.1 項（1-7）では、オペレーティング・システム・ユーティリティなど、システム関連の問題を特定するのに役立つシステム・パフォーマンス・ユーティリティに関して説明します。◆
- 8.3 項（8-40）では、CPU リソースを評価する方法について説明します。

OpenVMS VAX OpenVMS Alpha

- OpenVMS のドキュメント・セットでは、パフォーマンスの問題を特定する手順について説明します。管理対象のシステム構成と作業負荷に対して最適なパフォーマンスを実現する手順についても説明します。◆

システム・パフォーマンス全体のチューニングには、システムのメモリー管理を慎重に分析するという 1 つの側面があります。詳細は、4.4.3 項「ワーキング・セット・クォータ・パラメータのチューニング」（4-189）と 8.2 項「メモリー・リソースの分析」（8-38）を参照してください。

1.2.2 OpenVMS システム・パラメータとプロセス・パラメータ

OpenVMS VAX OpenVMS Alpha

十分なシステム・リソース、優れたデータベース設計、適切な Oracle Rdb パラメータ設定を使用しても、データベース・パフォーマンスはさらにシステム自体のパフォーマンスに左右されます。Oracle Rdb をインストールする間に、『Oracle Rdb7 Installation and Configuration Guide』に示すガイドラインに従って、OpenVMS パラメータの値とプロセス・アカウント・クォータを設定する必要があります。

チューニングが必要な場合は、観察した動作の慎重な分析、またはシステム・パフォーマンス・モニターの実行結果に基づいて、変更するパラメータを最小限に絞ってください。システムを分析し、監視するためのツールについては、1.3.1 項（1-7）を参照してください。パラメータは、システム・パラメータか特定のユーザーのユーザー認証ファイル（UAF）内のエントリです。システム・パラメータは、AUTOGEN コマンド・プロシージャを使用して変更します。AUTOGEN の特徴の 1 つは、変更に関連するパラメータを自動的に調整することです。UAF の値を制御するには、OpenVMS 認証ユーティリティ（AUTHORIZE）を使用します。システム・パラメータ、UAF ファイル、AUTOGEN、認証ユーティリティの詳細は、OpenVMS のドキュメント・セットを参照してください。

4.4.2 項（4-182）と 4.4.4 項（4-190）では、システム・パラメータとユーザー・アカウント・クォータの設定について詳しく説明します。◆

1.2.3 データベース設計

データベースの論理設計は、他の全パフォーマンスの要因の基盤になります。論理設計が適切でないと、オペレーティング・システム・パラメータや Oracle Rdb パラメータなど、他のパラメータを調整しても最適なパフォーマンスは得られません。論理設計が未熟な場合は、実質的にデータベースの分析、設計、構造化、ロードのすべてをやり直す必要があります。データベース・パフォーマンス改善の出発点として、管理者はデータに関する理解を深めるとともに、『Oracle Rdb7 Guide to Database Design and Definition』に示す設計の概念を熟知している必要があります。

データベースの物理設計の改善（チューニング）には、データ記憶域と検索を左右する要因の分析と、その要因を制御するパラメータの調整があります。パラメータの分析と調整については、それぞれ 3 章（3-1）と 4 章（4-1）を参照してください。詳細は、8 章（8-1）「データベース・リソースのボトルネックの診断」を参照してください。Oracle Trace と Oracle Expert for Rdb[*1] の使用も検討してください。上記の 2 製品の簡単な説明は、1.3.1 項（1-7）を参照してください。

1.2.4 アプリケーション設計

データベース・パフォーマンスに悪影響をおよぼす最後の要素は、アプリケーションの設計です。次に示すデータベース・プログラミングに関する一般的なガイドラインに従って、パフォーマンスを最適化し、競合を削減してください。

- トランザクションを短くし、ロックを削減するとともにスナップショット・ファイルの拡張を抑えます（可能な場合）。
- 複雑なクエリーを複数の単純なクエリーに分割し、ビューを使用してプログラム・アクセスを標準化することで、トランザクションを単純にします。以上の手順によって次の効果が得られます。
  - クエリーの最適化に関するオーバーヘッドが軽減されます。
  - プログラミングおよび保守の問題発生の可能性が低下します。
- RESERVING 句で SHARED READ を指定して、ロックを最小限に抑え、同時アクセスを促進します。
- トランザクション内部での端末 I/O を回避し、断続的なロックの問題を回避します。
- オプティマイザを支援します。詳細は、5.8 項（5-37）を参照してください。
- dbkey を使用してデータ行に直接アクセスします（可能な場合）。これでインデックスの使用を回避し、ロックを最小限に抑えると同時に I/O も削減します。

1.2.5 パフォーマンス関連の変更

この項では、システムまたはデータベースに対して管理者が変更できる項目で、データベース・パフォーマンスを左右し得るものをリストアップします。

OpenVMS VAX OpenVMS Alpha

システムの変更には、LOCKIDTBL、REHASHTBL、GBLPAGES、GBLSECTIONS、MAXBUF、VIRTUALPAGECNT、DEADLOCK_WAIT など、OpenVMS パラメータの調整があります。Oracle Rdb アプリケーションは上記のパラメータを使用して最適な設定を決定します。

[*1] Oracle Trace と Oracle Expert for Rdb は、OpenVMS システム以外では使用できません。

プロセスの変更には、ENQLM、WSQUO、WSDEF、WSMAX、FILLM、BYTLM、ASTLM、DIOLM、BIOLM など、OpenVMS パラメータの調整があります。Oracle Rdb アプリケーションは上記のパラメータを使用して最適な設定を決定します。◆

データベースの変更には、次のオプションがあります。

- データの分散
  ディスク・デバイス全体にデータベース・ファイルを再配分します。
  記憶領域全体に行を再配分し、より均一に分散するようにします。また、記憶領域全体で行を水平にパーティション化し、アクセスの多い行をグループ化します。
  同じ複合記憶領域内の 2 つ以上のテーブルをクラスタ化し、そのテーブルの行が同時にバッファに読み込まれるようにします。クラスタ化が不可能な場合は、シャドウ・ページの実装を検討します。
  データの冗長性を制御して検索パフォーマンスを改善します（結合が少なくて済む）。ただし、更新のパフォーマンスは低下します。

- インデックス
  複合記憶領域にハッシュ・インデックスを追加します。
  データベースの最も重要な用途（クエリーまたは更新トランザクション）に基づいて、インデックスの特性（ノード・サイズ、使用率、使用方法オプション）を設定します。
  データベースに行をロードしてからインデックスを定義します。データベースに行をロードする前に、主キーでソートします。

- ロック
  論理領域内の行の競合が非常に激しく、CPU 時間が問題になる場合は、ALG を無効にします。また、変更することのない読取り / 書込み記憶領域を読取り専用記憶領域に変更し、行ロックの競合をさらに削減します。更新の少ないテーブルに対してデータ圧縮を使用します。
  他のデータベース・ユーザーとのロック競合の可能性を低下する手段として、トランザクション・スコープを最適化します。
  スナップショットを有効または無効にし、更新ユーザーと検索ユーザーとのロックの競合を削減します（遅延スナップショット・ファイルの使用も検討します）。

- I/O
  データベースに関する主要なトランザクション・タイプのバッファ・サイズ、バッファの数、バッファ・プールを最適化します。
  主要なトランザクション・タイプによって、グローバルなバッファまたは高速コミット処理あるいはその両方を有効にします。

- データベース・パラメータ
  様々な記憶領域のページ・サイズを最適化し、行断片化を回避します。
  様々なページ割当て範囲とページ・ブロック・サイズを指定して、どの構成で最も均一な行分散が得られるかを判断します。
  均一記憶領域と複合記憶領域に対して、SPAM しきい値と SPAM ページ間のデータ・ページ間隔を設定します。
  IMPORT/EXPORT 操作を実行し、複合記憶領域の拡張をなくします。

1.3 パフォーマンス・ユーティリティおよびツール

データベース環境のパフォーマンスを評価するために、ツールとユーティリティが用意されています。たとえば、OpenVMS ユーティリティ、Oracle RMU コマンド、Oracle Rdb の論理名と構成パラメータなどがあります。1.3.1 項（1-7）から 1.3.3 項（1-10）では、各ツールとユーティリティを紹介し、説明の参照先を示します。

1.3.1 オペレーティング・システム・ユーティリティ

OpenVMS VAX OpenVMS Alpha

オペレーティング・システムに対してデータベース管理システムが大きな影響を与えることから、データベースのパフォーマンスは、オペレーティング・システム全体のパフォーマンスと密接に関連しています。

OpenVMS オペレーティング・システムは、データベースのパフォーマンスを、OpenVMS オペレーティング・システム環境の全体において評価するために、次のユーティリティを提供しています。

- DCL MONITOR コマンドは、システム・リソースの使用状況と統計を継続的に表示します。
- DCL SHOW コマンドは、システム・リソース使用率のスナップショットを表示します。
- DCL DUMP コマンドは、ファイルやボリュームの内容を、ASCII、10 進、16 進、8 進で表示または印刷します。
- SYSGEN コマンドは、現行の OpenVMS システム・パラメータを表示します。
- AUTOGEN コマンドは、OpenVMS システム・パラメータを変更できます。
- AUTHORIZE コマンドは、ユーザー定義パラメータを表示し、変更できます。

OpenVMS のユーティリティとコマンドの詳細は、OpenVMS のドキュメントを参照してください。

次の Oracle 製品は、システム・パフォーマンスを監視するのに有効です。

- Oracle Trace のソフトウェア
  Oracle Trace は、OpenVMS レイヤー製品と Oracle Rdb アプリケーションの任意の組合せから収集したイベント・ベースのデータに基づいて、データ収集や報告を行う製品です。Oracle Rdb では、実行時に発生する多くのイベントが事前に定義されています。イベントには、開始と終了があるもの（トランザクションの開始からコミットまで）と、発生するだけのもの（データベースにアタッチ）があります。Oracle Trace では、この事前に定義された Oracle Rdb のイベントとデータ項目を関連付けることができます。
  標準のリソース使用率に関する項目や、Oracle Rdb を使用したアプリケーションに固有の項目を関連付けることができます。たとえば、イベント名やイベントが発生するプロシージャなど、イベント自体に関する情報も指定できます。
  ワーキング・セットのサイズ、イベント発生時刻など、プロセス統計やパフォーマンス

情報も指定できます。アプリケーションで Oracle Trace Call（2.4.1 項（2-75）を参照）を使用すると、Oracle Trace によってアプリケーションからイベント・データを収集できます。このイベント・データは、次のように様々な目的に使用できます。

- アプリケーション・パフォーマンスのチューニング
- ハードウェア・リソースの計画（容量計画）
- データベース・パフォーマンスのチューニング
- アプリケーションのデバッグ
- エラーのロギング
- アプリケーション・パフォーマンス・データを Oracle Expert for Rdb にインポート

Oracle Trace は、イベント・ベースという点で他のデータ収集製品とは異なります。他の多くの製品は時間ベースです。**イベント・ベースのデータ収集製品**は、プログラム・コード内で事前に定義した場所で、そのコードが実行された場合にデータを収集します。**時間ベースのデータ収集製品**は、指定した間隔でデータを収集します。イベント・ベースのデータ収集製品には、次のような利点があります。

- 固有レポートの生成やトランザクションなど、アプリケーション内の特定のイベントで実際に使用するリソースに関するデータを簡単に収集し、報告できます。
- トランザクションを実行する頻度などについて、平均や推定でなく実際の頻度を測定します。

Oracle Trace は、アプリケーションやデータベースのパフォーマンスの分析や変更はできません。Oracle Trace の機能は、ユーザーが要求するデータを収集し、そのデータに基づいてレポートを作成することです。作成されたレポートを解析するのは、ユーザーや他のレイヤーの製品です。Oracle Trace を Oracle Rdb アプリケーションで使用する方法については、2.4 項（2-74）を参照してください。Oracle Trace の使用方法に関する詳細は、Oracle Trace のマニュアルを参照してください。

- Oracle Expert for Rdb ソフトウェア
  Oracle Expert for Rdb は、データベースを設計し、実装するためのソフトウェア • ツールです。Oracle Expert for Rdb には、物理的なデータベース設計原理やデータベース管理の特質（Oracle Rdb Query Optmizer など）に基づくチューニング規則があります。設計を行う場合には、Oracle Expert for Rdb でデータベースの論理的な設計、テーブルとインデックスのカーディナリティ、トランザクションの作業負荷情報をインポートします。この場合は、作業負荷の優先順位とデータベース環境情報を入力する必要があります。特定の設計を選択した理由を説明するレポートが生成されます。Oracle Trace で収集した作業負荷のデータは、Oracle Expert for Rdb に直接インポートできます。
  情報の分析が終わると、Oracle Expert for Rdb はデータベースの作成に使用するコマンド・ファイルを生成します。Oracle Expert for Rdb では、データベースをロードおよびアンロードするプロシージャも生成できます。
  Oracle Trace と Oracle Expert for Rdb を使用して、既存のデータベースをチューニング

することもできます。Oracle Trace を使用して稼働中のシステムに関するデータを収集し、このデータを Oracle Expert for Rdb にインポートします。次に、Oracle Expert for Rdb を使用して既存のデータベースをアンロードし、設計を変更してから再ロードします。詳細は、『Oracle Expert for Rdb User's Guide』を参照してください。◆

1.3.2 Oracle RMU のコマンド

Oracle RMU（Rdb Management Utility）を使用すると、Oracle Rdb データベースに関する情報を表示できます。データベース・パフォーマンスの分析に特に有効な Oracle RMU コマンドには、RMU Analyze、RMU Show、RMU Dump があります。

RMU Analyze

データベース・パフォーマンスの問題の主な原因は、ストレージが足りないことです。RMU Analyze コマンドを使用すると、データベースのスペースの使用を監視できます。データベースのスペースの使用を定期的に分析することで、問題を早期に発見でき、実際に問題が発生する前に対応を開始できます。2 章（2-1）では、RMU Analyze コマンドとその修飾子を紹介します。RMU Analyze の具体的な例とその出力は、このマニュアルの 3 章と 4 章で示します。

RMU Show

Oracle Rdb は、データベース全体の様々なデータベース・アクティビティに関する統計を保持しています。この統計を使用すると、個々のデータベース領域の I/O 処理によるプログラムのデータベース・アクセス効率を評価できます。この統計は、インデックス・ノードの評価、全データベース・アクセス・パターンの評価、ロック統計評価に使用できます。

2 章（2-1）では、RMU Show Statistics コマンドを使用して Performance Monitor を起動し、データベース統計を表示する方法を説明します。Performance Monitor 画面については、このマニュアル全体を通して説明します。

RMU Show Locks コマンドは、特定のノード上に配置されたアクティブな全データベースのプロセス・ロックに関する最新の情報を表示します。個々のプロセスまたは一連のプロセスで使用するロックのタイプがわかれば、どのプロセスが他のプロセスをブロックしているかを確認できます。RMU Show Locks コマンドの詳細は、3.8.1.1 項（3-44）を参照してください。

RMU Dump

パフォーマンス評価とデータ配分の問題、特に行の配置の分野では、ディスク上のデータベースの内容を調べるのにこのコマンドが役立ちます。場合によっては、データベースの記憶領域と論理領域について、内部のルート・ファイルの情報と ID 番号を確認する必要があります。記憶領域と論理領域の情報を表示し、領域管理（SPAM）ページ、領域インベントリ・ページ（AIP）、領域ビット・マップ（ABM）、ユーザーのインデックス構造とデータ行を格納するデータ・ページを調べることもできます。RMU Dump コマンドを使用してデータベース・ファイルの内容を表示し、解析する方法は、『Oracle Rdb7 Guide to Database

Maintenance』を参照してください。RMU コマンドの詳細は、『Oracle RMU Reference Manual』を参照してください。

1.3.3 Oracle Rdb の論理名と構成パラメータ

Oracle Rdb では多くの論理名と構成パラメータをユーザーが定義できるので、複数のパラメータをプロセスごとに制御することでパフォーマンスを向上できます。たとえば、Oracle Rdb の論理名と構成パラメータによって、.ruj ファイルを再配置または事前拡張することや、実行時にユーザーが割り当てるバッファの番号を一時的に変更することができます。他の Oracle Rdb 論理名と構成パラメータは、オプティマイザ・ストラテジを検討するのに役立ちます。

表 2-6（2-61）に、すべての Oracle Rdb 論理名と構成パラメータを示し、それぞれの機能を簡単に説明します。付録 A に、個々の論理名と構成パラメータに関する詳しい説明があります。

1.4 パフォーマンス分析の方法

データベース・パフォーマンスの評価を開始する前に、ハードウェアのリソースやオペレーティング・システムのリソースの不足や不良など、データベース・パフォーマンス低下の原因になり得る要素を排除します。ハードウェアのリソースやオペレーティング・システムのリソースを確認する前に、データベース環境を評価するのは避けてください。システム構成の変更やシステムの作業負荷の大幅な変更によって、システム全体のパフォーマンスが変化する可能性があることを忘れないでください。

データベース・パフォーマンスに関する考慮事項の多くは、個々のアプリケーションの属性によって異なります。設計、分析、チューニング、プログラミングに絶対的な規則はありません。また、どんなデータベース・パフォーマンスの問題にも適用できる解決策はありません。データベース・パフォーマンスは、データベースの使用目的と実際の使用状況に基づいて評価する必要があります。パフォーマンスの問題は、データベース設計が実際の使用状況に適合しない場合に発生します。

この項の残りの部分では、データベース・パフォーマンスを分析する場合の推奨事項を示します。具体的な問題の評価に役立つステップごとの詳細な説明については、8 章（8-1）のデシジョン・ツリーも参照してください。

1.4.1 チューニングのガイドライン

データベース・パフォーマンス分析の最初のステップは、本当に問題が存在するのかを判断することです。非現実的な期待から、パフォーマンスに関する不満が発生している場合もあります。問題があると判断した場合は、次の一般的な質問に答えてください。

- 最初に問題が発生したのはいつですか。
- 問題のある動作は断続的ですか。それとも継続していますか。
    - 1 日の特定の時間に発生頻度が上がりますか。

- 特定のタイプのトランザクションで発生していますか。
- 特定のユーザーや特定のプログラムで問題の発生頻度が上がりますか。
- 最後に問題が発生したのはいつですか。同じシステム上で他に発生している問題がありますか。

- 最初に問題が発生する直前になんらかの変更を行いましたか。最近、新しいソフトウェアかハードウェアをインストールしましたか。
- 最近、作業負荷の量など、環境上の要因に変更がありましたか。
- サード・パーティ製品をデータベースと併用していますか。

情報収集を開始する際は、以上の質問に回答することが効果的です。

1.4.2 チューニングの効果を解析するためのコンテキストの確立

データベース・パフォーマンスを分析するためには、評価の結果を解析するコンテキストを確立する必要があります。目標を設定し、その達成度を評価します。たとえば、平均応答時間を 10% 短縮するという目標を設定します。ページのロックを有効にしてパーティション・アプリケーションのレコード・ロックのオーバーヘッドを除去するなど、パフォーマンス改善の効果を測定します。

データベース・パフォーマンスの評価と改善を開始する場合に、測定可能な条件で達成すべき目標がわからなければ、パフォーマンスが改善されてもそれで問題が解決するか判断できません。個々の変更の結果を測定すれば、個々の変更によって改善の程度がわかります。変更とそれによって得られたパフォーマンス改善のパーセンテージのログを保存すれば、同じ領域が原因で繰り返し問題を発生しているかどうかがわかります。

パフォーマンスの問題が非常に顕著な場合は、チューニングを行ってください。これで、問題の原因を特定しやすくなります。

1.4.3 テスト・データベースの使用

テスト・データベースは、データベースとそれにアクセスするプログラムのパフォーマンスの評価に有効です。実際的なテスト環境を作成し、パフォーマンス・テストだけでなくデータベースの変更に関するテストを行ってから本番データベースに実装してください。

データベースが大きすぎて正確なコピーを作成できない場合は、たとえば 1/10 の縮小版を作成します。本番データベースのデータ構成と同じパーセンテージのテスト・データで構成した、テスト・データベースをロードします。ページ・サイズが一致していることを確認します。ストレージ・ハッシング・アルゴリズムではページ数の相違によって若干異なる結果が得られることがあるので、結果がまったく同じとは言えませんが、10 分の 1 に縮小したデータベースでも同じような結果になります。

テスト・データベースで同じ問題を再現できたことを確認します。必要に応じて、テスト・データベースに対して本番プログラムを実行します。空き領域があれば、本番ディレクトリ構造とまったく同じテスト・ディレクトリ構造をセットアップします。この作業には、専用

ディスクを 1 つか 2 つ用意することをお薦めします。適切なテスト環境を作成するための投資は、ただちに本番データベースの改善という効果で還元され、本番データベース自体をチューニングする危険性が低下します。

1.4.4 データベースの変更

データベースのパフォーマンスを左右するパラメータを変更する場合は、次の 2 つの規則に従います。

- 一回に 1 項目の変更
- 簡単な順に変更

1.4.4.1 1 項目の変更

テスト・データベースとそれにアクセスするプログラムの変更は、一回に 1 項目ずつ行います。個々の変更の効果に注目し、変更がデータベース環境に与えた影響を評価します。変更によってパフォーマンスが低下する場合は、データベース環境を元の状態に戻します。

テスト・データベースを変更し、それを確認したら、その変更を本番データベースに実装する計画を立てます。

1.4.4.2 簡単な順に変更

データベース・パフォーマンスに関連する変更は、実装しやすい順に行ってください。

- 最も実装しやすい変更を最初に行います。たとえば、After-image ジャーナル・ファイルの割当てや拡張のように、オンライン（ユーザーがデータベースに接続されている場合やトランザクションを実行中の場合）で変更でき、ただちに効果が得られる場合があります。
- オンラインで変更はできても、新しいトランザクションが発生するまで、あるいはユーザーがデータベースから切断し、データベースに再接続するまで効果がない場合もあります。たとえばこのような変更には、記憶領域が読取り専用か読取り / 書込みかを示す、読取り専用フラグの設定があります。
- データベースに接続できるユーザー数の変更など、オフライン（ユーザーがデータベースに接続されていない場合）で変更でき、データベースのアンロードや再ロードが必要ない場合もあります。
- 記憶領域の拡張や再編成など、オフラインで変更し、データベースのアンロードや再ロードが必要な場合もあります。

4.1 項（4-2）に、データベース・パラメータの調整方法に関する説明があります。パフォーマンスの問題のうち、データベース全体にわたる特定のパラメータと記憶領域に関するパラメータの指定変更に関連するものの多くは、次のいずれかの方法で解決できます。

- SQL の ALTER DATABASE 文を使用して、データベース全体を適切に変更します。

- SQL の ALTER DATABASE 文を使用して、新しい記憶領域を定義し、対応するストレージ・マップ関連の文を変更して、自動的に指定したテーブルの行を新しい記憶領域に移動します。

オンラインとオフラインで実行できるデータベース変更（データベースのエクスポートとインポートを行わないもの）の詳細な説明は、『Oracle Rdb7 Guide to Database Maintenance』を参照してください。『Oracle Rdb7 Guide to Database Maintenance』には、SQL の EXPORT 文と IMPORT 文を使用してデータベースをエクスポートおよびインポートする手順の説明もあります。

1.4.5 パフォーマンス評価手順の例

この項では、オンライン・データベース・アプリケーションのパフォーマンスを評価する手順の概要を示します。

1. SQL の EXPORT 文と IMPORT 文を使用してデータベースを新しいディスクのロケーションに移動し、データ・リポジトリ（Oracle CDD/Repository[*1]）情報を新しいリポジトリ・ディレクトリに移動します。
2. データベースにアクセスするアプリケーション・プログラム・コードのセクションを特定します。
3. アプリケーションの実際の動作を再現するプログラム・テスト・モジュールを作成します。

OpenVMS VAX OpenVMS Alpha

4. OpenVMS ランタイム・ライブラリ・コール LIB$INIT_TIMER と LIB$SHOW_TIMER か Oracle Trace を使用して、経過時間や CPU 時間などの OpenVMS オペレーティング・システム統計を生成します。◆
5. Performance Monitor を使用して、読み込まれたデータベースのページ数、ロック・アクティビティ、I/O 統計など、Oracle Rdb 統計を監視します。
6. 作業負荷手順（一定の間隔によるプログラム・モジュールの実行やバッチ処理によるオンライン・ユーザーのシミュレーション）を確立します。
7. テスト・データベースを変更し、変更の前後の統計出力を比較します。

1.4.6 クラスタ・パフォーマンスに関する考慮事項

クラスタ環境におけるデータベース・パフォーマンスは、データベース・ファイルをディスク上のどこに配置するかによって変わることがあります。たとえば、大規模なクラスタ環境で Oracle Rdb を使用する場合は、クラスタ環境ではルートファイルの I/O アクティビティが増大するため、データベース・ルート・ファイルを単独の専用ディスクに配置しようとすることがあります。

[*1] OpenVMS システムのみ

RMU Restore コマンドや RMU Move_Area コマンドを使用してデバイスの仕様を変更し、頻繁に使用する大規模な記憶領域を多くのディスクに分散する場合もあります。ファイルへの様々なディスク・アクセス・パスを試すこともできます。たとえば、HSC ディスクはシークと回転に関して最適化されているので、HSC 上のファイルは MASSBUS ディスクと UNIBUS ディスク上のデータベース・ファイルよりも、高速にアクセスできることがわかります。

OpenVMS VAX OpenVMS Alpha

VMScluster 環境における Oracle Rdb パフォーマンスの詳細は、6 章（6-1）を参照してください。◆

2

データベース・パフォーマンス分析ツール

この章では、Oracle Rdb データベースのパフォーマンスを分析するためのツールについて説明します。次のツールがあります。

- RMU Analyze コマンド
- Performance Monitor
- Oracle Rdb の論理名と構成パラメータ
- Oracle Trace for OpenVMS
- Oracle Expert for Rdb ◆

OpenVMS VAX OpenVMS Alpha

この章では、Oracle RMU コマンドおよび Oracle Rdb の論理名と構成パラメータの概要を示します。各ツールの使用方法は、このマニュアルの該当する項を参照してください。たとえば、RMU Analyze Indexes コマンドの説明は、3.9.5 項（3-97）にあります。Oracle Trace の説明は、2.4 項（2-74）にあります。

2.1 RMU Analyze コマンド

RMU Analyze コマンドを使用すると、次のデータベース特性を調べることができます。

- データベースで使用する領域、記憶領域内の論理領域で領域を利用する方法（ページ・レベル）、各ページの空き領域
- 領域割当て、ページ・サイズ、現行のデータベース・パラメータ、SPAM のしきい値と間隔、ページ・フォーマット、行圧縮、一般的な行配置の情報など、データベース定義をサポートする現行のデータベース・パラメータ、およびこれらのパラメータが機能する効率
- データベースが現在保持するレコードの数とタイプ、そのページ上の位置、行断片化と行圧縮に関する統計
- インデックスの構造、レベルの数、重複するノードの数、一意キーの数、および個々のインデックスとノード・レコードに関する詳細情報（実際の dbkey、レベル、サイズ、キーの長さ、実際のキー値など）
- インデックスと特定のインデックスに対応するテーブル上のデータ・レコードの配置、データ行に達するインデックス・レコードの数、データ行に達するまでに横断するページの最大数、データ行にアクセスするのに必要なデータベース・バッファの数
- 次の各パラメータのヒストグラム（頻度分布）
  - アクセスするレコード数
  - 横断するページ数
  - 使用するバッファ数

  データ行の頻度分布については、次の詳細情報があります。
  - データ行にアクセスするのに必要な dbkey の数
  - データ行に達するまでにアクセスするページの数
  - データ行とインデックス構造がバッファ内に存在するかどうか
  - キーの長さ
  - 実際のキー値

RMU Analyze コマンドは、次の作業に有効です。

- 断片化やページ・オーバーフローが発生した場合に、SQL の ALTER DATABASE 文を使用してデータベース・パラメータをリセットします。
- データベースのエクスポートとインポートの手順を作成します。
- インデックスを定義します。

- 新しい記憶領域を追加し、1 つ以上のテーブルの行をその新しい領域に分割し、I/O のボトルネックを解消します。
- 新しい記憶領域を追加し、記憶域マップの変更に基づいて元の記憶領域の行を再編成します。

Binary_Output 修飾子を使用すると、RMU Analyze コマンドの結果をバイナリ・ファイルに書き込むことができます。

RMU Analyze コマンドを使用するには、RMU 権限の RMU$ANALYZE、または OpenVMS 権限の SYSPRV か BYPASS が必要です。

2.1.1 RMU Analyze コマンド修飾子

RMU Analyze コマンド出力によって、データベースの空き状況、行断片化の有無、行の分散状況、ページ上のレコード型、データ行に達するまでの I/O パスの長さがわかります。以上のような情報は、データベースがどのように展開しているかを認識し、データベース・パフォーマンスをどのように改善するかを決定するのに役立ちます。

この項では、RMU Analyze コマンドを表 2-1（2-3）に示す 4 つの修飾子を付けて使用した場合に収集できる情報の概要を説明します。表 2-1（2-3）には、各修飾子の詳細と例の参照先も示します。

表 2-1 RMU Analyze コマンド修飾子

修飾子	参照先
Areas	4.2.1.1 項（4-120）
Lareas	4.2.1.2 項（4-135）
Indexes	3.9.5.1 項（3-97）
Placement	4.3.1.1 項（4-158）～ 4.3.1.3 項（4-166）

個々の修飾子による詳細出力レベルは、Option={Normal | Full | Debug} 修飾子を使用して変更できます。

- Normal
  サマリー情報のみを出力します（デフォルト）。
- Full
  ヒストグラムとサマリー情報を出力します。
- Debug
  ヒストグラムとサマリー情報のほかに、データに関する内部情報を出力します。

次の各 RMU Analyze コマンド修飾子の下にリストアップされる表示要素に特に注意することで、データベース・パフォーマンスを改善できます。

- Areas
  - 記憶領域のページ使用状況
    記憶領域のページ領域使用率とページ数のヒストグラムの形状を調べます。使用率が 60% 未満のページが多い場合は、ランダムな削除を何度も実行しているか、一般的なサイズの行を保持するにはページの空き領域が小さすぎる可能性があります。たとえば、ページ・サイズが 2 ブロック（1024 バイト）で一般的な行が 550 バイト（圧縮後）の場合、Oracle Rdb では 1 ページに 1 行が格納されます。したがって、ページあたり 474 バイトが未使用になります。7 行を格納する場合は、7 ページすなわち 7168 バイトを必要とし、そのうちの 3318 バイトは未使用になります（7 × 474 = 3318）。
    しかし、ページ・サイズが 4096 バイトの場合は、Oracle Rdb では 7 行すべてをちょうど 1 ページに格納できます。ページ・サイズが 1024 バイトの場合は 3318 バイトが未使用になるのに対して、この場合は 246 バイトだけが未使用になります。
  - SPAM カウント
    SPAM ページの数は、記憶領域にほとんど空きがない場合に、空き領域を検出するのに必要な I/O 操作の最大数を近似します。
  - オーバーフロー
    オーバーフローは、行断片化とハッシュ・バケット・オーバーフローによって発生する可能性があります。行断片化は、記憶領域のページ・サイズの見積りが小さすぎるために発生することがあります。圧縮特性が圧縮から非圧縮に変更された場合に、ページに非圧縮行を格納できる領域がないために発生することもあります。
    ハッシュ・インデックス・レコード（ハッシュ・バケットと重複するノード・レコード）とデータ行のオーバーフローは、ページに格納しなければならないすべてのレコードに比べて、ページ・サイズが小さすぎる場合に発生します。この原因としては、重複するレコードの分散を評価しなかったか、ページ上のハッシュ・バケットのサイズを低く見積もったことが考えられます（特に、PLACEMENT VIA INDEX オプションを使用してデータ行とハッシュ構造の両方を同じページに格納した場合）。
- Lareas
  - 断片化
    行断片化は、RMU Analyze コマンドの Area 修飾子と Lareas 修飾子に対応する出力のサマリーで示されます。表示される情報は、断片化された行の数、各行内の断片の数、断片化された各行のバイト数です。断片化された行が多いと、パフォーマンスが低下します。
  - 論理領域のページ領域使用率とページ数
    論理領域のページ領域使用率とページ数のヒストグラムの形状を調べます。このヒストグラムは、特定の論理領域のデータに使用するページ領域のパーセンテージに対して、データを格納するページの数を示します。ヒストグラムの形状から、各ページに行がどのように分散しているかがわかり、データベースが変化する様子についてなんらかの情報が得られます。

    - 論理領域の最大レコード長に対するパーセンテージとレコード数
      最大レコード長に対する論理領域のパーセンテージとレコード数のヒストグラム（Option=Full 修飾子で生成される）の形状を調べます。このヒストグラムは、各レコードのレコード型の最大サイズ（非圧縮）に対する論理領域のパーセンテージを示します。このヒストグラムの形状によって、論理領域のレコード長の分布がわかります。

- Indexes
    - インデックス・レベル
      I/O 操作の数を使用して検索のコストを評価します。
    - インデックス・ノードの数とサイズ
      インデックスの使用に関連するストレージ・オーバーヘッドを評価します。
    - 重複するインデックスの数とサイズ
      インデックスの効率を評価します。一意のインデックスは重複値の多いインデックスより高速です。重複値の多いインデックスを使用すると、隣接するページにオーバーフローすることがあります。オーバーフローは、ページ・サイズが小さすぎるためにページがインデックス・レコードでいっぱいになり、隣接するページにはみ出した場合に発生します。
- Placement
    - 最大と平均のパスの長さ
      データ行に達するまでにアクセスするインデックス・レコードの最大と平均の数を示します。
    - 最大 I/O パスの長さ
      データ行に達するまでに横断するページの最大数を示します。
    - 最小 I/O パスの長さ
      特定のバッファ・サイズのデータ行にアクセスするのに必要なバッファの数を示します。
    - dbkey パスの長さと頻度
      dbkey パスの長さと頻度のヒストグラムの形状を調べます。このヒストグラムは、データ行に達するまでにアクセスするインデックス・レコードの数を示します。特定のインデックスについて、ヒストグラムの形状から、インデックスを定義したテーブルのすべてのデータ行に関する dbkey パスの長さの頻度分布がわかります。
    - 最大 I/O パスの長さと頻度
      最大 I/O パスの長さと頻度のヒストグラムの形状を調べます。このヒストグラムは、データ行に達するまでに横断するページの合計数を示します。特定のインデックスについて、ヒストグラムの形状から、インデックスを定義したテーブルの各データ行に達するまでに横断するページの合計数の頻度分布がわかります。
    - 最小 I/O パスの長さと頻度
      最小 I/O パスの長さと頻度のヒストグラムの形状を調べます。このヒストグラム

は、バッファ・サイズを考慮した場合に、データ行にアクセスするのに必要なバッファの数を示します。特定のインデックスについて、ヒストグラムの形状から、インデックスを定義したテーブルのデータ行にアクセスするのに必要なバッファの数の頻度分布がわかります。

- データ・ページ上のデータ行の分布に関する詳細情報
  データ・ページ上の特定のデータに達するまでに必要な dbkey の数、そのデータ行に達するまでの最大と最小の I/O パスの長さ、各キーの長さ、データ行に固有のキーを示します。

RMU Analyze コマンドに Areas 修飾子と Lareas 修飾子を付けて使用すると、トランザクションは開始されませんが、物理領域を分析する間、その領域を同時読込みに関してロックします。一度に 1 つのデータベース記憶領域に対して分析を行い、記憶領域ごと、論理領域ごとにレポートを生成します。Oracle Rdb は、各テーブルのデータごと、各テーブルのインデックスごとに別々の論理領域を保守し、セグメント文字列データのデータベースごとに 1 つの論理領域を保守します。RMU Analyze コマンドでは、レコードのロックもスナップショット・ファイルも使用しないので、同時にデータベース・アクティビティを実行できます。

一般に、データベース領域の読取りや更新の必要な他のすべてのトランザクション（読取り専用、読取り / 書込み共有読取り、読取り / 書込み保護付き読取り、読取り / 書込み共有書込み、読取り / 書込み保護付き書込み）は、Areas 修飾子と Lareas 修飾子を指定した場合に RMU Analyze コマンドとは競合しません。記憶領域の情報を読み込むのに、スナップショット・ファイルを有効にする必要はありません。ただし、実行中のデータベース・アクティビティによって、RMU Analyze コマンドの結果は時間の経過に従って変化する場合があります。

RMU Analyze Indexes コマンドか RMU Analyze Placement コマンドを使用すると、読取り専用トランザクションが開始されます。情報を収集するには、スナップショット・ファイルを有効にする必要があります。これで、RMU Analyze Indexes コマンドか RMU Analyze Placement コマンドを使用する間に、他のデータベース・アクティビティを実行できます。競合が発生する可能性があるのは、インデックス分析や配置分析を実行中の物理領域に対して排他的なトランザクションやバッチの更新トランザクションを実行しようとした場合に限ります。これは、いずれのトランザクションもスナップショット・ファイルに Before-image を書き込まないためです。

2.1.1.1 RMU Analyze コマンド出力からの Oracle Rdb 情報の除外

RMU Analyze、RMU Analyze Indexes、RMU Analyze Placement コマンドの出力から、Oracle Rdb 情報を除外することができます。

RMU Analyze コマンドで、Exclude=System_Records 修飾子か Exclude=Metadata 修飾子を指定します。Exclude=System_Records 修飾子を指定すると、RDB$SYSTEM_RECORD 論理領域に関する情報が RMU Analyze の出力から除外されます。Exclude=Metadata 修飾子を指定すると、すべての Oracle Rdb 論理領域（RDB$SYSTEM_RECORD 論理領域や

RDBVMS$COLLATIONS_NDX 論理領域など）に関する情報が RMU Analyze の出力から除外されます。

RMU Analyze Indexes コマンドか RMU Analyze Placement コマンドで Exclude=Metadata 修飾子を指定すると、すべての Oracle Rdb インデックス（RDB$NDX_REL_NAME_NDX インデックスや RDBVMS$COLLATIONS_NDX インデックスなど）に関する情報が RMU Analyze Indexes と RMU Analyze Placement の出力から除外されます。

データは Exclude 修飾子で除外された論理領域についても収集されますが、コマンド出力からは除外されます。

2.1.2 後から分析に使用するためのバイナリ出力ファイルの作成

RMU Analyze コマンドには非常に便利なオプションがあり、バイナリ・ファイルすなわち固定長レコードのバイナリ・ファイル（.unl）か Oracle CDD/Repository[*1] 互換のレコード定義ファイル（.rrd）に結果を出力できます。このファイルは、次のように後続の分析に使用できます。

- RMU Load Rms_Record_Def コマンドでサマリーの結果を Oracle Rdb データベースにロードすると、ユーザーが記述した管理のアプリケーションやプロシージャで使用できます。
- ユーザーが記述したプログラムからこの結果にアクセスできます。

次のいずれかのコマンドで Binary_Output 修飾子を指定すると、バイナリ出力ファイルを作成できます。

- RMU Analyze（Areas 修飾子か Lareas 修飾子を指定）
- RMU Analyze Indexes
- RMU Analyze Placement

例 2-1（2-7）に示すように、Binary_Output= (File = file-spec) 修飾子を指定すると、固定長レコードのバイナリ・ファイル（.unl ファイル・タイプ）を作成できます。

例 2-1 固定長レコードのバイナリ・ファイルの作成

```
$ RMU/ANALYZE/BINARY_OUTPUT=FILE=ANALYZE_OUT1
```

例 2-2（2-7）に示すように、Binary_Output= (Record_Definition = file-spec) 修飾子を使用すると、Oracle CDD/Repository 互換レコード定義バイナリ・ファイル（.rrd ファイル・タイプ）を作成できます。

例 2-2 Oracle CDD/Repository 互換レコード定義バイナリ・ファイルの作成

```
$ RMU/ANALYZE/BINARY_OUTPUT=RECORD_DEFINITION=ANALYZE_OUT2
```

[*1] OpenVMS システムのみ

例 2-3（2-8）に示すように、2 つの修飾子をカンマで区切ってカッコで囲むと、同じコマンド行で両方のバイナリ・ファイルを作成できます。

例 2-3 固定長レコード・ファイルと Oracle CDD/Repository 互換レコード定義ファイルの両方の作成

```
$ RMU/ANALYZE/BINARY_OUTPUT=(FILE=ANALYZE_OUT1,RECORD_DEFINITION=ANALYZE_OUT2)
```

ただし、デフォルトは Nobinary_Output であり、バイナリ・ファイルは作成されません。

OpenVMS VAX OpenVMS Alpha

例 2-4（2-8）に示す RMU Analyze コマンドでは、データ・リポジトリと互換性のある DB.RRD という RMS レコード定義ファイルに結果を出力します。

例 2-4 RMU Analyze による Oracle CDD/Repository 互換レコード定義ファイルの作成

```
$ RMU/ANALYZE/BINARY_OUTPUT=RECORD_DEFINITION=DB.RRD mf_personnel
$!
$! Display the DB.RRD file created by the previous command:
$ TYPE DB.RRD
DEFINE FIELD RMU$DATE DATATYPE IS DATE.
DEFINE FIELD RMU$AREA_NAME DATATYPE IS TEXT SIZE IS 32.
DEFINE FIELD RMU$STORAGE_AREA_ID DATATYPE IS SIGNED WORD.
DEFINE FIELD RMU$FLAGS DATATYPE IS SIGNED WORD.
DEFINE FIELD RMU$TOTAL_BYTES DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$EXPANDED_BYTES DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$FRAGMENTED_BYTES DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$EXPANDED_FRAGMENT_BYTES DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$TOTAL_COUNT DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$FRAGMENTED_COUNT DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$FRAGMENT_COUNT DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$PAGE_LENGTH DATATYPE IS SIGNED WORD.
DEFINE FIELD RMU$MAX_PAGE_NUMBER DATATYPE IS SIGNED LONGWORD.
DEFINE FIELD RMU$FREE_BYTES DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$OVERHEAD_BYTES DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$AIP_COUNT DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$ABM_COUNT DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$SPAM_COUNT DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$INDEX_COUNT DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$BTREE_NODE_BYTES DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$HASH_BYTES DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$DUPLICATES_BYTES DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$OVERFLOW_BYTES DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$LOGICAL_AREA_ID DATATYPE IS SIGNED WORD.
DEFINE FIELD RMU$RELATION_ID DATATYPE IS SIGNED WORD.
DEFINE FIELD RMU$RECORD_ALLOCATION_SIZE DATATYPE IS SIGNED WORD.
DEFINE FIELD RMU$TOTAL_SPACE DATATYPE IS F_FLOATING.
DEFINE RECORD RMU$ANALYZE_AREA.
    RMU$DATE.
    RMU$AREA_NAME.
```

```
    RMU$STORAGE_AREA_ID.
    RMU$FLAGS.
    RMU$TOTAL_BYTES.
    RMU$EXPANDED_BYTES.
    RMU$FRAGMENTED_BYTES.
    RMU$EXPANDED_FRAGMENT_BYTES.
    RMU$TOTAL_COUNT.
    RMU$FRAGMENTED_COUNT.
    RMU$FRAGMENT_COUNT.
    RMU$PAGE_LENGTH.
    RMU$MAX_PAGE_NUMBER.
    RMU$FREE_BYTES.
    RMU$OVERHEAD_BYTES.
    RMU$AIP_COUNT.
    RMU$ABM_COUNT.
    RMU$SPAM_COUNT.
    RMU$INDEX_COUNT.
    RMU$BTREE_NODE_BYTES.
    RMU$HASH_BYTES.
    RMU$DUPLICATES_BYTES.
    RMU$OVERFLOW_BYTES.
    RMU$LOGICAL_AREA_ID.
    RMU$RELATION_ID.
    RMU$RECORD_ALLOCATION_SIZE.
    RMU$TOTAL_SPACE.
END RMU$ANALYZE_AREA RECORD.
```

表 2-2（2-9）に、DB.RRD レコードの各フィールドを示します。

表 2-2 フィールドの説明

フィールド	定義
RMU$ABM_COUNT	記憶領域内の ABM ページの数
RMU$AIP_COUNT	記憶領域内の AIP ページの数
RMU$AREA_NAME	分析した記憶領域の名前
RMU$AVAILABLE	インデックス・レコード内の最初の空き領域のサイズ
RMU$BTREE_NODE_BYTES	記憶領域内でソートされたインデックスのバイト数
RMU$COUNT	インデックス・ノードの数
RMU$DATA_COUNT	レコードの数
RMU$DATE	分析操作を実行した日付

表 2-2 フィールドの説明（続き）

フィールド	定義
RMU$DUPLICATES_BYTES	記憶領域内でソートされたインデックスの重複するキー値のバイト数
RMU$DUPLICATE_AVAILABLE	重複するレコード内の最初の空き領域のサイズ
RMU$DUPLICATE_COUNT	重複するレコードの数
RMU$DUPLICATE_DATA_COUNT	重複するレコードの数
RMU$DUPLICATE_KEY_COUNT	重複するキーの数
RMU$DUPLICATE_USED	重複するレコード内の使用可能な領域の使用済みサイズ
RMU$EXPANDED_BYTES	論理領域に格納されたデータの解凍後の合計サイズ
RMU$EXPANDED_FRAGMENT_BYTES	格納された断片の解凍後のバイト数
RMU$FLAGS （DB.RRD レコードの場合）	このフィールドの 3 つの可能な値には次の意味がある。 ■ 3 - 論理領域のデータ圧縮が有効である。 ■ 1 - 論理領域のデータ圧縮が有効でない。 ■ 0 - レコードが記憶領域レコードであり、論理領域レコードでない。

表 2-2 フィールドの説明（続き）

フィールド	定義
RMU$FLAGS（INDEX.RRD レコードと PLACEMENT.RRD レコードの場合）	このフィールドの 8 つの可能な値には次の意味がある。 ■ 7 - インデックスがハッシュされており、一意である。詳細なレポートが生成される。 ■ 6 - インデックスがハッシュされており、一意ではない。詳細なレポートが生成される。 ■ 5 - インデックスがソートされており、一意である。詳細なレポートが生成される。 ■ 4 - インデックスがソートされており、一意ではない。詳細なレポートが生成される。 ■ 3 - インデックスがハッシュされており、一意である。詳細なレポートは生成されない。 ■ 2 - インデックスがハッシュされており、一意ではない。詳細なレポートは生成されない。 ■ 1 - インデックスがソートされており、一意である。詳細なレポートは生成されない。 ■ 0 - インデックスがソートされており、一意ではない。詳細なレポートは生成されない。
RMU$FRAGMENTED_BYTES	格納された断片のバイト数
RMU$FRAGMENTED_COUNT	断片化されたレコードの数
RMU$FRAGMENT_COUNT	格納された断片の数
RMU$FREE_BYTES	記憶領域内の空きバイト数
RMU$HASH_BYTES	記憶領域内のハッシュ・インデックスのバイト数
RMU$INDEX_COUNT	記憶領域内のインデックス・レコードの数
RMU$INDEX_NAME	分析したインデックスの名前
RMU$KEY_COUNT	キーの数
RMU$LEVEL	インデックス・レベルの最大数
RMU$LOGICAL_AREA_ID	分析した論理領域の論理領域 ID
RMU$MAX_KEY_PATH	任意のレコードにアクセスするまでにアクセスするキーの最大数
RMU$MAX_PAGE_NUMBER	記憶領域内で最後に初期化されたページのページ番号

表 2-2 フィールドの説明（続き）

フィールド	定義
RMU$MAX_PAGE_PATH	任意のレコードにアクセスするまでにアクセスするページの最大数
RMU$MIN_BUF_PATH	任意のレコードにアクセスするまでにアクセスするバッファの最小数
RMU$OVERFLOW_BYTES	記憶領域内のハッシュ・バケット・オーバーフロー・レコードのバイト数
RMU$OVERHEAD_BYTES	記憶領域内でオーバーヘッドに使用されたバイト数
RMU$PAGE_LENGTH	記憶領域内のデータベース・ページのバイト長
RMU$RECORD_ALLOCATION_SIZE	テーブルを最初に定義したときの行のサイズ
RMU$RELATION_ID	分析した論理領域内の行のレコード型
RMU$RELATION_NAME	インデックスを定義するテーブルの名前
RMU$SPAM_COUNT	記憶領域内の SPAM ページの数
RMU$STORAGE_AREA_ID	分析した記憶領域の領域 ID
RMU$TOTAL_BUFFER_PATH	すべてのレコードにアクセスするまでにアクセスするバッファの合計数
RMU$TOTAL_BYTES	論理領域に格納されたデータの合計サイズ
RMU$TOTAL_COMPRESSED_IKEY_COUNT	圧縮された（レベル 1）インデックス・キーが占めるバイト数の合計
RMU$TOTAL_COUNT	格納されたレコードの合計数
RMU$TOTAL_IKEY_COUNT	圧縮が有効でない場合にインデックスによって消費されたバイト数の合計
RMU$TOTAL_KEY_PATH	すべてのレコードにアクセスするまでにアクセスするキーの合計数
RMU$TOTAL_PAGE_PATH	すべてのレコードにアクセスするまでにアクセスするページの合計数
RMU$TOTAL_SPACE	論理領域内でユーザー・データの格納に使用できるバイト数（使用済み領域 + 空き領域 + オーバーヘッド）
RMU$USED	使用可能な領域の使用済みサイズ

例 2-5（2-13）に示す RMU Analyze Index コマンドでは、リポジトリと互換性のある INDEX.RRD という RMS レコード定義ファイルに結果を出力します。

例 2-5 RMU Analyze Index による Oracle CDD/Repository 互換レコード定義ファイルの作成

```
$ RMU/ANALYZE/INDEX/BINARY_OUTPUT=RECORD_DEFINITION=INDEX.RRD mf_personnel
$!
$! Display the INDEX.RRD file created by the previous command:
$ TYPE INDEX.RRD
DEFINE FIELD RMU$DATE DATATYPE IS DATE.
DEFINE FIELD RMU$INDEX_NAME DATATYPE IS TEXT SIZE IS 32.
DEFINE FIELD RMU$RELATION_NAME DATATYPE IS TEXT SIZE IS 32.
DEFINE FIELD RMU$LEVEL DATATYPE IS SIGNED WORD.
DEFINE FIELD RMU$FLAGS DATATYPE IS SIGNED WORD.
DEFINE FIELD RMU$COUNT DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$USED DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$AVAILABLE DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$DUPLICATE_COUNT DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$DUPLICATE_USED DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$DUPLICATE_AVAILABLE DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$KEY_COUNT DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$DATA_COUNT DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$DUPLICATE_KEY_COUNT DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$DUPLICATE_DATA_COUNT DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$TOTAL_COMPRESSED_IKEY_COUNT DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$TOTAL_IKEY_COUNT DATATYPE IS F_FLOATING.
DEFINE RECORD RMU$ANALYZE_INDEX.
    RMU$DATE.
    RMU$INDEX_NAME.
    RMU$RELATION_NAME.
    RMU$LEVEL.
    RMU$FLAGS.
    RMU$COUNT.
    RMU$USED.
    RMU$AVAILABLE.
    RMU$DUPLICATE_COUNT.
    RMU$DUPLICATE_USED.
    RMU$DUPLICATE_AVAILABLE.
    RMU$KEY_COUNT.
    RMU$DATA_COUNT.
    RMU$DUPLICATE_KEY_COUNT.
    RMU$DUPLICATE_DATA_COUNT.
    RMU$TOTAL_COMPRESSED_IKEY_COUNT.
    RMU$TOTAL_IKEY_COUNT.
END RMU$ANALYZE_INDEX RECORD.
```

表 2-2（2-9）に、INDEX.RRD レコードの各フィールドを示します。

例 2-6（2-14）に示す RMU Analyze Placement コマンドは、リポジトリと互換性のある PLACEMENT.RRD という RMS レコード定義ファイルに結果を出力します。

例 2-6 RMU Analyze Placement による Oracle CDD/Repository 互換レコード定義ファイルの作成

```
$ RMU/ANALYZE/PLACEMENT/BINARY_OUTPUT=RECORD_DEFINITION=PLACEMENT.RRD -
_$ mf_personnel
$!
$! Display the PLACEMENT.RRD file created by the previous command:
$ TYPE PLACEMENT.RRD
DEFINE FIELD RMU$DATE DATATYPE IS DATE.
DEFINE FIELD RMU$INDEX_NAME DATATYPE IS TEXT SIZE IS 32.
DEFINE FIELD RMU$RELATION_NAME DATATYPE IS TEXT SIZE IS 32.
DEFINE FIELD RMU$LEVEL DATATYPE IS SIGNED WORD.
DEFINE FIELD RMU$FLAGS DATATYPE IS SIGNED WORD.
DEFINE FIELD RMU$COUNT DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$DUPLICATE_COUNT DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$KEY_COUNT DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$DUPLICATE_KEY_COUNT DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$DATA_COUNT DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$DUPLICATE_DATA_COUNT DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$TOTAL_KEY_PATH DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$TOTAL_PAGE_PATH DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$TOTAL_BUFFER_PATH DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$MAX_KEY_PATH DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$MAX_PAGE_PATH DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$MIN_BUF_PATH DATATYPE IS F_FLOATING.
DEFINE RECORD RMU$ANALYZE_PLACEMENT.
    RMU$DATE.
    RMU$INDEX_NAME.
    RMU$RELATION_NAME.
    RMU$LEVEL.
    RMU$FLAGS.
    RMU$COUNT.
    RMU$DUPLICATE_COUNT.
    RMU$KEY_COUNT.
    RMU$DUPLICATE_KEY_COUNT.
    RMU$DATA_COUNT.
    RMU$DUPLICATE_DATA_COUNT.
    RMU$TOTAL_KEY_PATH.
    RMU$TOTAL_PAGE_PATH.
    RMU$TOTAL_BUFFER_PATH.
    RMU$MAX_KEY_PATH.
    RMU$MAX_PAGE_PATH.
    RMU$MIN_BUF_PATH.
END RMU$ANALYZE_PLACEMENT RECORD. ◆
```

表 2-2（2-9）に PLACEMENT.RRD レコードの各フィールドの説明があります。

[No]Binary_Output 修飾子の構文と使用方法の詳細は、『Oracle RMU Reference Manual』を参照してください。データを Oracle Rdb データベースにロードする RMU Load Rms_Record_Def コマンドの使用方法の詳細は、『Oracle Rdb7 Guide to Database Design and Definition』を参照してください。

2.2 Performance Monitor

Oracle Rdb は、繰り返し発生する特定のデータベース・アクティビティに関する統計情報を収集して保持します。Performance Monitor を使用すると、データベース統計を表示して、パフォーマンスに関する種々の問題領域を識別できます。その後、SQL の ALTER DATABASE 文を使用して、データベースを変更できます。

2.2.1 データベース統計の使用方法

データベース統計は、Oracle Rdb を実行している各コンピュータ・システム上のグローバル・セクションまたは共有メモリー・パーティションに保持されています。クラスタ環境では、Oracle Rdb は各ノードで別々に統計を保持しています。このような環境では、データベース統計は、Performance Monitor を起動したクラスタ内のノードから発生したアクティビティのみを反映します。

RMU Show Statistics コマンドを使用して、Performance Monitor を実行します。データベース統計は、次に示す時点まで Performance Monitor によって収集されます。

- データベースが閉じられるまで。
  データベースは、最後のプロセスがデータベースの接続を解除したときに閉じられます。Performance Monitor プロセスは、データベースを開いたままにします。データベースを閉じると、データベース統計は 0（ゼロ）にリセットされます。
- Until 修飾子で指定した時間まで。
  RMU Show Statistics コマンドを発行したときに Until 修飾子を使用して時間を指定した場合、統計収集は指定した時刻に終了します（データベースをあらかじめ閉じる必要はありません）。RMU Show Statistics コマンドの構文の詳細は、『Oracle RMU Reference Manual』を参照してください。
- Reset オプションは、画面上の水平メニューか、RMU Show Statistics コマンドラインの Reset 修飾子を使用して指定できます。水平メニューの Unreset オプションは、Reset オプションとは逆の働きをします。
  Reset は、現在行っている統計の収集を停止し、新しい統計の収集を開始します。

処理アクティビティを観察するためにデータベースを開いている時間を延長する場合、RMU Open コマンドを使用して、明示的にデータベースを開くことができます。

論理名 RDM$BIND_STATS_ENABLED または構成パラメータ RDB_BIND_STATS_ENABLED を使用すると、プロセスに対するデータベース統計の書出しを無効にすることができます。デフォルトでは、データベース統計の書出しは、ノード上の各プロセスに対して有効です。すなわち、RDM$BIND_STATS_ENABLED または RDB_BIND_STATS_ENABLED の値は、すべてのプロセスに対して 1 または真に設定されます。RDM$BIND_STATS_ENABLED 論理名または RDB_BIND_STATS_ENABLED 構成パラメータを使用してプロセスに対するデータベース統計を無効にすると、Performance Monitor はそのプロセスに対する統計を収集しません。画面上に表示された統計には、統計が無効であるプロセスの情報は含まれていません。すでにチューニングされており、Performance Monitor が提供する情報を必要としない、静的かつパフォーマンスに対する要求が厳しいアプリケーションにとっては、統計を無効にすることが便利です。

RDM$BIND_STATS_ENABLED または RDB_BIND_STATS_ENABLED の値を 0（ゼロ）に設定することで、プロセスに対するデータベース統計の書出しを無効にすることができます。次の例は、OpenVMS 上のデータベース統計を無効にする方法を示しています。

```
$ DEFINE RDM$BIND_STATS_ENABLED 0
```

データベース統計の収集が無効であるプロセスについてデータベース統計の書出しを有効にするには、値に 1 を指定した論理名または構成パラメータを定義するか、論理名または構成パラメータの割当てを解除します。次の例は、OpenVMS でデータベース統計を有効にする方法を示しています。

```
$ ! Set the RDM$BIND_STATS_ENABLED value to 1:
$ DEFINE RDM$BIND_STATS_ENABLED 1
$ !
$ ! Or, deassign the logical name:
$ DEASSIGN RDM$BIND_STATS_ENABLED
```

すべてのプロセスに対する統計の書出しは、次のいずれかの方法を使用して無効にできます。

- 各プロセスに対して、RDM$BIND_STATS_ENABLED 論理名または RDB_BIND_STATS_ENABLED 構成パラメータを 0（ゼロ）に設定します。RDM$BIND_STATS_ENABLED をグループ論理またはシステム論理として定義すると、この作業が簡単になります。◆

  OpenVMS VAX OpenVMS Alpha

- データベース自体に対する統計収集を無効にします。データベースに対する統計収集が無効の場合、そのデータベースにアタッチされているプロセスに対する統計も表示されません。SQL CREATE DATABASE 文、ALTER DATABASE 文または IMPORT 文の STATISTICS COLLECTION IS DISABLED 句を使用すると、データベースに対する統計収集を無効にできます。統計収集を無効または有効にするには、SQL 構文を使用するようお薦めします。次の ALTER DATABASE 文は、mf_personnel データベースに対する統計収集を無効にします。

  ```
  SQL> ALTER DATABASE FILENAME mf_personnel
    1> STATISTICS COLLECTION IS DISABLED;
  ```

データベースに対する統計収集を有効にするには、SQL CREATE DATABASE 文、ALTER DATABASE 文または IMPORT 文の STATISTICS COLLECTION IS ENABLED 句を使用します。次の ALTER DATABASE 文は、mf_personnel データベースに対する統計収集を有効にします。

```
SQL> ALTER DATABASE FILENAME mf_personnel
  1> STATISTICS COLLECTION IS ENABLED;
```

デフォルトでは、統計収集はデータベースに対して有効です。

次に示す統計情報を表示できます。

- ディスク I/O 操作
- メモリー使用量
- 待ち時間を含むロック
- インデックス・アクティビティ
- トランザクション継続時間
- スナップショット・ファイルの使用方法
- バッファ・アクティビティ
- 行キャッシュ・アクティビティ
- プロセス情報
- データベース・パラメータ情報
- After-image ジャーナルのオーバーヘッド
- 現行ノードにおける After-image ジャーナル

次の形式で統計を表示できます。

- グラフィック表示
- 数値表示
- 個々のフィールドのタイム・プロット（秒単位のトランザクションなど）
- 散布図プロット表示
- 表による表示

表示形式の詳細は、2.2.6.1 項（2-26）から 2.2.6.5 項（2-35）を参照してください。

2.2.2 RMU Show Statistics コマンドの構文

RMU Show Statistics コマンドを使用して、Performance Monitor の文字セル・インタフェースを起動します。このコマンドの基本構文は、次のとおりです。

```
RMU Show Statistics [ database-file-name ]
```

RMU Show Statistics コマンド修飾子の説明は、『Oracle RMU Reference Manual』を参照してください。

2.2.3 文字セル・インタフェースで表示モードを選択

Performance Monitor は、オンライン、レコード、再生の 3 つのモードで動作します。

- オンライン
  オンライン・モードでは、データベース・アクティビティは、発生するたびに監視されます。デフォルトでは、Oracle RMU は 3 秒ごとに新しい統計を記録します。
  Performance Monitor を起動するときに、Time 修飾子で適切な間隔を指定できます。また、インタラクティブ・セッションの間は、いつでも間隔を変更できます。メニューから「Set_rate」を選択して、適切な間隔を指定します。
  Interactive 修飾子を使用すると端末に統計を表示できます。また、Output 修飾子を使用するとバイナリ・ファイルに統計を記録できます。この両方を行うこともできます。
  Oracle Rdb は、通常、データベースが開いている限りデータベースに対する統計値を収集します。Until 修飾子を使用すると、統計収集を終了する時刻を指定できます。Until 修飾子は、通常バッチ・ジョブで Output 修飾子とともに使用されます。

- レコード
  Output 修飾子を使用して、統計値をバイナリ・ファイルに記録できます。このバイナリ・ファイルは、印刷して読めるリストではありません。バイナリ・ファイルを読み出すには、Replay モードを使用する必要があります。RMU Show Statistics コマンドを使用し、Input 修飾子の引数としてバイナリ・ファイルの名前を指定します。ファイルのフォーマットを変更するプログラムを作成することもできます。バイナリ・ファイルの詳細は、2.2.13 項（2-52）を参照してください。

- 再生
  Replay モードでは、以前バイナリ・ファイルに記録されたデータベース・アクティビティを読み込んで端末に表示します。この統計は、Output 修飾子を使用して、オンライン・モードで記録されたものの可能性もあります。
  以前に記録した統計を再生するには、Input 修飾子を使用して、その統計を含むファイルの名前を指定します。次に、矢印キーとメニューを使用して、オンライン・モードと同じように表示形式や画面を変更します。Replay モードを使用すると、統計をサンプリングして何度でも表示でき、またサンプリングを様々な視点で検査できます。これは、ある画面で何かを見つけたときに、別の画面を表示して疑問を確認する場合に有効です。
  Time 修飾子で希望する割合を指定するか、または「Set_rate」メニューのオプションを使用して表示率を変更できます。Replay モードでの表示率の変更は、オンライン・モードでの変更とは異なります。たとえば、デフォルト間隔の 3 秒を使用して、オンライン・モードでファイルに統計を収集したとします。このファイルから統計を再生する場合、表示率を変更すると、統計自体が変更されるのではなく、3 秒間隔で表示される

表示率が変更されます。
「Select Input Control」メニューを使用すると、Replay モードでのデータベース統計の表示を制御できます。詳細については、2.2.4 項（2-19）を参照してください。

Nointeractive 修飾子を使用すると、インタラクティブな表示を抑制できます。これは、後で参照するためのバイナリ統計値出力ファイルを生成する場合で、統計を収集する間はオンライン表示しない場合に便利です。

いくつかの統計は、データベース・ルート・ファイルのアクティビティを監視します。マルチファイル・データベースの場合、.rdb ファイルは、ユーザー・データではなく、データベース・ヘッダー情報のみを含むデータベース・ルート・ファイルです。

2.2.4 文字セル・インタフェースにおける「Select Input Control」メニューの使用

Performance Monitor が Replay モードの場合、「I」と入力して現在の画面の水平メニューから「Input」オプションを選択します。「Input オプション」を選択すると、Replay モードに対する「Select Input Control」メニューが表示されます。

```
Select Input Control
A. Replay
B. Pause
C. Continue
D. Place Markpoint
```

次に、使用できるメニュー・オプションとその意味を示します。

- A. Replay
  入力ファイルを最初から再生します。これは、Replay モードと Pause モードのいずれかから選択できます。「Replay」オプションを選択した場合、統計は入力ファイルの最初から表示されます。
- B. Pause
  現在の表示を休止します。「Pause」オプションを選択すると、Performance Monitor は Pause モードになります（表示ヘッダーの「Mode:」フィールドが Replay から Paused に変わることに注意してください）。「Select Input Control」メニューから「Continue」オプションを選択するまで、表示は Pause モードのままです。「Pause」オプションを選択すると、Pause モードに対する「Select Input Control」メニューが表示されます。このメニューでは、オプション「B」は「Single-Step」です。

```
Select Input Control
A. Replay
B. Single-Step
C. Continue
D. Place Markpoint
```

Pause モードでは、現在の画面に対して、2 つの新しいオプション（「Continue」と「Single-Step」）が水平メニューに表示されます。
画面に対するすべての水平メニューは、「Set_rate」を除いて Pause モードでも通常どおりに機能します（「Set_rate」オプションを使用して表示率を変更できますが、統計自体は変更されません）。Pause モードでは、たとえば表示形式を数値形式からグラフィック形式に、またその逆に変更できます。このオプションと「Write」オプションは、何を出力しているのか正確にわかるので特に便利です。画面が休止状態になったときに山カッコ（[>]）キーと（[<]）キーを使用して、使用可能なページ間を移動することもできます。これを使用すると、最適な瞬間の真のスナップショットを得ることができます。
Pause モードの間に入力ファイルを 1 レコード進めるには、「Select Input Control」メニューからオプション「B」（Single-Step）を選択するか、[S] を入力して水平メニューから「Single-Step」を選択します。Pause モードでは、Replay モード（「Select Input Control」メニューのオプション「A」）を選択し、「Single-Step」オプションを使用して、入力ファイルのレコード全体またはレコードの一部を実行することができます。

- C. Continue
  前に選択した割合で画面の通常表示を再開します。水平メニューで [C] を入力して「Continue」オプションを選択すると、Performance Monitor は Replay モードに戻ります。Replay モードを再開すると、「Continue」オプションと「Single-Step」オプションが水平メニューから削除され、「Set_rate」オプションに置き換えられます。また、「Select Input Control」メニューは Replay モードオプションのみの表示に戻ります。

- D. Place Markpoint
  これを使用すると、入力ファイルの現在のレコードをマークできるので、後でそのレコードにすぐに戻ることができます。Oracle RMU は、そのレコードがマークポイントとして入力ファイルに作成された時刻（この時刻はディスプレイ・ヘッダーの一番上に表示されています）を使用します。このオプションを選択すると、新しいオプションが「Select Input Control」メニューに追加されます。

```
      Select Input Control
A. Replay
B. Pause
C. Continue
D. Place Markpoint
E. Goto Markpoint 11:17:59
```

- E. Goto Markpoint <time>
  これを使用すると、前に保存したマークポイントにすぐに戻ります。マークポイントは、メニュー・オプションの一部として表示されている時刻で識別できます。オプション「E」を選択すると、そのマークポイントまで進むかまたは戻ります。
  マークポイントを取り除くことはできませんが、「Select Input Control」メニューから再度「D」を選択することで新しいマークポイントを設定できます。マークポイントを使用すると、特定の意味を持つ位置を識別できます。現在、同時に設定できるマークポ

イントは 1 つだけです。「Goto Markpoint」オプションを選択すると、統計は入力ファイル内のマークポイントから表示されます。

2.2.5 Performance Monitor でのナビゲート

データベース・アクティビティに関して多くの異なる統計を表示する画面を選択するには、Performance Monitor が提供するメニューを使用します。Performance Monitor を起動すると、最初の画面が表示されます。最初の画面の最下行に水平メニューがあります。たとえば、次のように表示されます。

```
--------------------------------------------------------------------------------
Exit Graph Help Menu Options Pause Reset Set_rate Time_plot Unreset Write X_plot
--------------------------------------------------------------------------------
```

各画面には水平メニューがあり、各メニュー選択の最初の文字がハイライトされています。そのメニューでハイライトされている文字を入力して、選択します。[M] を入力すると、使用可能な画面のメニューが表示されます。

```
                                Select Display
   A. Summary IO Statistics              O. IO Statistics (by file)  [->
   B. Summary Locking Statistics         P. Locking (one lock type)  [->
   C. Summary Object Statistics          Q. Locking (one stat field) [->
   D. Summary Cache Statistics           R. Lock Statistics (by file)[->
   E. Summary Cache Unmark Statistics    S. Database Parameter Info  [->
   F. Record Statistics                  T. Row Cache (One Cache)    [->
   G. Transaction Duration (Total)       U. Row Cache (One Field)    [->
   H. Custom Statistics                  V. Row Cache Information    [->
   I. Snapshot Statistics                W. Index Information        [->
   J. Process Information       [->      X. General Information      [->
   K. Journaling Information    [->      Y. Objects (one stat type)  [->
   L. Hot Standby Information   [->      Z. Objects (one stat field) [->
   M. IO Statistics             [->      0. Database Dashboard       [->
   N. Global Buffer Information[->       1. Online Analysis & Info.  [->
```

次の方法で画面を選択できます。

- 選択する画面の文字を入力します。
- 矢印キーを使用してカーソルを移動し、選択する画面をハイライトします。[Enter] キーを押します。
  [↑] キーと [↓] キーを使用してメニュー内を上または下に移動できます。[←] キーと [→] キーを使用して、メニュー内の 2 つの列間を移動できます。
- プラス（+）キーとマイナス（-）キーを使用します。
  プラス・キーは画面内を前に移動し、マイナス・キーは画面内を後に移動します。プラス・キーまたはマイナス・キーを入力すると、Performance Monitor は、前に進むまたは後ろに進む画面数の入力を要求します。Performance Monitor は指定した画面の数だ

け移動します。[+0] を入力すると、Performance Monitor は最後の画面に移動します。[-0] を入力すると、Performance Monitor は最初の画面に移動します。

- ツールまたはヘルプ機能の「Goto」画面オプションを使用します。
  これらのオプションの説明は、表 2-4（2-38）を参照してください。

ある画面を表示していて、表示メニューやツール類を使用せずに別の画面に移動する場合は、[←] キーまたは [Prev Screen] キー、および [→] キーまたは [Next Screen] キーを使用して移動することができます。[←] キーまたは [Prev Screen] キーを押すと前の画面に移動します。[→] キーまたは [Next Screen] キーを押すと次の画面に移動します。画面の順序は、「Select Display」メニューで示されている順序です。すなわち、たとえば、「Transaction Duration」画面（「Select Display」メニューで「G」を選択）を表示している場合、[←] キーを一度押すと「Record Statistics」画面（「F」を選択）に移動できます。また、[→] キーを一度押すと「Custom Statistics」画面（「H」を選択）に移動できます。

画面には、1 ページより長いものもあります。表示ヘッダーは、画面のページ数を示しています。山カッコ（[>]）キーを押すと、水平メニューに >「next_page」オプションとして示されているように、続きのページを表示します。そして、前のページに戻るには、水平メニューに <「prev_page」オプションとして示されているように、山カッコ（[<]）キーを押します。山カッコ（[>]）キーは、表示の最後のページで停止し、山カッコ（[<]）キーは最初のページで停止します。

[↑] キーおよび [↓] キーを使用して、複数のページを含む表示を移動することもできます。矢印キーを使用すると、循環方式で移動できる利点が加わります。表示の最初のページが現在の画面の場合、[↑] キーを押すと表示の最後のページに移動します。最後のページで [↓] キーを押すと、表示の最初のページに移動します。

「Select Display」メニューのオプションの後に [-> があれば、そのオプションを選択すると、Performance Monitor は、サブメニューを表示することを意味しています。

たとえば、「IO Statistics (by file)」表示を選択すると、「Select File」メニューが表示されます。

```
                Select File
        A. File IO Overview
        B. Device IO Overview
        C. Device Information
        D. root file
        E. AIJ file
        F. RUJ file
        G. ACE file
        H. all data/snap files
        I. data file MF_PERS_DEFAULT
        J. data file EMPIDS_LOW
        K. data file EMPIDS_MID
        L. data file EMPIDS_OVER
        M. data file DEPARTMENTS
        N. data file SALARY_HISTORY
        O. data file JOBS
        P. data file EMP_INFO
        Q. <<more>>
```

I/O 統計を表示するデータベースを選択するには、対象ファイルの文字を入力するか、カーソルをそのファイルに移動して選択し、[Enter] キーを押します。すべてのデータベース・ファイルを 1 つの画面で表示できない場合、メニューの最初の画面で「<<more>>」オプション（Q）を選択し、追加の「Select File」メニュー選択を表示できることに注意してください。

「Select Display」、「Select File」、「Select Lock Type」および「Select Lock Field」メニューで、次のキーを押すと以下のように応答します。

- [↑]
  メニュー選択リストを上方向に移動します。
- [↓]
  メニュー選択リストを下方向に移動します。
- [←]
  2 列のメニューで、左の列に移動します。1 列のメニューでは何も起こりません。
- [→]
  2 列のメニューで、右の列に移動します。1 列のメニューでは何も起こりません。
- OpenVMS では [Ctrl]+[W]、Compaq Tru64 UNIX では [Ctrl]+[L]
  画面のリフレッシュ
- OpenVMS では [Ctrl]+[Z]、Compaq Tru64 UNIX では [Ctrl]+[D]
  メニューの取消

水平メニューで、各メニュー選択の最初の文字がハイライトされます。水平メニューから選択するには、ハイライトされている文字を入力します。表 2-3（2-24）は、使用可能（画面による）なメニュー選択と選択後の応答を示しています。

表 2-3 Performance Monitor 水平メニュー選択肢

Alarm	この時間を超えてプロセスが停止すると「Stall Messages」画面に表示される、プロセスの停止継続時間を指定できます。
Bell	Alarm ベルを有効または無効にします。
Brief	画面の概要を表示します。全画面バージョンと概要バージョンの両方を持つ画面でのみ使用できます。
Config	統計画面の構成を行います。
Continue	前に選択した割合で画面の通常表示を再開します。
Exit	Performance Monitor を終了します。水平メニューで OpenVMS では [Ctrl]+[Z]、Compaq Tru64 UNIX では [Ctrl]+[D] を押しても、Performance Monitor を終了できます。
Filter	失効メッセージをフィルタできます。
Full	全画面バージョンで画面を表示します。全画面バージョンと概要バージョンの両方を持つ画面でのみ使用できます。
Graph	この画面の統計値をグラフィック形式で表示します。
Help	キーボードの使用法、および現在の画面とそのフィールドに関するヘルプを起動します。
Input	「Select Input Control」メニューを表示します。
LockID	Lock ID のサブメニューを表示します。
Menu	「Select Display」メニューを表示します。
Normal	通常の画面表示に戻ります。「Time Plot」表示または「Scatter Plot」表示を入力する前に表示されていた画面を表示します。
Numbers	この画面の統計を数値（グラフ）形式で表示します。
Options	デフォルト・ディレクトリにある STATISTICS.RPT ファイルに、特定の時刻に対する画面を書き出せるようにします。このファイルを、グラフィック形式、数値形式、または両方の形式で書き出すことができます。
Pause	画面の出力を休止できます。「Pause」メニュー・オプションがある画面のみ休止できます。他の画面は継続して更新され、休止できません。[P] キーをもう一度押すと、休止を解除します。
Refresh	画面のリフレッシュ

表 2-3 Performance Monitor 水平メニュー選択肢（続き）

Reset	データベース統計値を 0（ゼロ）にリセットします。
Set_rate	収集間隔を指定した間隔に変更します。
Step	Pause モードで入力ファイルを 1 レコード進めます。
Time_plot	あるフィールドに対するイベントの件数を詳細な図式で表示します。
Unreset	「Reset」オプションと反対の働きをします。
Update	データベース属性の値を変更できます。
Write	ヘルプを除く任意の画面の内容を、デフォルト・ディレクトリの RMU.SCR ファイルに書き出します。
X_plot	特定のフィールドに対する垂直ヒストグラムを提供します。
Yank	カスタム統計を表示するために選択できる統計フィールドのメニューを表示します。選択したフィールドは「Custom Statistics」画面に自動的に表示されます。
Zoom	特定の項目に対する詳細な情報を表示します。
>Next_page	[→] キーを押すと、現行画面の次のページに移動します。
<Prev_page	[←] キーを押すと、現行画面の前のページに移動します。
!	ツール機能を起動して、「Select Tool」サブメニューを表示します。
$	OpenVMS で DCL コマンドを実行します。
#	現行画面のサブメニューを表示します。
+	n 画面前に進めます。
--	n 画面後ろに戻ります。

どのデータベース・アクティビティが発生しているかいつも明らかであるとは限らないので、しばしばアクティビティが表示されている「Performance Monitor」画面を迅速に探すことが必要な場合があります。[space] バーを使用すると、現在アクティブであるサブメニューで次のデータ画面を検索できるようになります。現行のサブメニューにアクティビティを持つ画面がなければ、Performance Monitor は [Next Screen] キーを使用した場合と同様に次の画面を表示します。また、計算結果画面（「Row Cache Status」など）および情報画面（「Stall Messages」、「Monitor Log」、または「AIJ Journal Information」画面など）は、アクティブ・データの検索中は無視されます。検索モードは、バイナリ入力ファイルの再生中にも使用できます。

2.2.6 文字セル・インタフェースで表示書式を選択

Performance Monitor が収集した情報は、次の形式で表示できます。

- Graph
- Numbers
- Time Plot
- Scatter Plot
- Table

Graph 表示形式は、統計を画像で表示します。したがって、データベースがどのように実行されているかすぐに理解できます。Numbers 表示形式は、実際に収集された数値をチャートで表示します。Time Plot 表示形式は、画面によっては使用できますが、特定の表示フィールドを繰り返し監視できます。Scatter Plot 表示形式は、画面によっては使用できますが、垂直ヒストグラム・フォームの特定のフィールドを現行の割合で監視できます。Table 表示形式は、表形式で統計を表示します。

各画面には、形式にかかわらず同じヘッダー情報が表示されます。

```
------------------------------------------------------------------------------
Node: TRIXIE          Oracle Rdb V7.0-00 Performance Monitor  2-MAY-1996 12:40:40
Rate: 3.00 Seconds            Summary IO Statistics          Elapsed: 00:00:29.05
Page: 1 of 1      RDBVMS_USER1:[LOGAN.V7.TMP]MF_PERSONNEL.RDB;1      Mode: Online
------------------------------------------------------------------------------
```

ヘッダーの最初の行には、ノード名、ユーティリティ名およびバージョン番号、現在のシステムの日付と時刻（指定した Set_rate 間隔で自動的に更新）が含まれています。2 番目の行には、秒単位で行われる画面リフレッシュ・レート、画面の名前、前回の Reset コマンドからの経過時間が記述されています。3 行目には、画面内での現在のページ番号、現在のデータベースの名前、Performance Monitor の表示モード（Online、Record または Replay）が記述されています。

2.2.6.1 項（2-26）から 2.2.6.5 項（2-35）では、表示形式を詳細に説明します。

2.2.6.1 グラフィック表示形式

グラフ表示形式は、トランザクション、Verb の成功と失敗、データベースの I/O アクティビティ、ある画面におけるその他のデータ特性などのイベントについて、1 秒間の発生レートを示します。グラフの各バーは、現在のイベント・レート、つまり直前のサンプル間隔における特定イベントの 1 秒間の発生レートを反映しています。次にグラフィック表示の例を示します。

```
Node: TRIXIE         Oracle Rdb V7.0-00 Performance Monitor 17-MAY-1996 14:10:24
Rate: 3.00 Seconds           Summary IO Statistics          Elapsed: 00:02:32.50
Page: 1 of 1          SQL$DISK1:[USER]MF_PERSONNEL.RDB;1            Mode: Online
------------------------------------------------------------------------------
statistic......... max. cur.          10        20        30        40        50
name.............. rate rate +---------+---------+---------+---------+---------+
                             |         |         |         |         |         |
transactions           1    0 |         |         |         |         |         |
verb successes        80   80 +---------+---------+---------+---------+--------->|
verb failures         11   11 +---------+*        |         |         |         |
                             |         |         |         |         |         |
synch data reads      17   17 +---------+------*  |         |         |         |
synch data writes      0    0 |         |         |         |         |         |
asynch data reads      0    0 |         |         |         |         |         |
asynch data writes     0    0 |         |         |         |         |         |
RUJ file reads         0    0 |         |         |         |         |         |
RUJ file writes        0    0 |         |         |         |         |         |
AIJ file reads         0    0 |         |         |         |         |         |
AIJ file writes        0    0 |         |         |         |         |         |
ACE file reads         0    0 |         |         |         |         |         |
ACE file writes        0    0 |         |         |         |         |         |
root file reads        7    0 |         |         |         |         |         |
root file writes       1    0 |         |         |         |         |         |
------------------------------------------------------------------------------
Exit Graph Help Menu Options Pause Reset Set_rate Time_plot Unreset Write X_plot
```

次に、グラフィック表示のレイアウトを説明します。

- statistic name
  この列は統計の名称を示します。表 2-5（2-47）に、各画面とその説明が記載されている場所を示します。
- max rate
  Performance Monitor が起動された後の、このイベントの 1 秒あたりの最大レートを示します。[R] を入力して「Reset」メニュー・オプションを選択するとこのカウンタをリセットできます。
- cur rate
  直前のサンプル間隔における、そのイベントの 1 秒間のレートを示します。RMU Show Statistics コマンドに Time 修飾子を指定すると、サンプル間隔をあらかじめ設定できます。また、[S] を入力して「Set_rate」オプションを選択すると表示した後でも変更できます。

グラフのバーは現在のレートを示していますが、グラフの幅の値は 50 に制限されています。現在のレートが 50 以下であれば、バーの最後にアスタリスク（*）が表示されます。現在のレートの値が 50 よりも大きければ、バーは山カッコ（>）で終わります。実際の値は「cur. rate」列に表示されます。

[No]Histogram 修飾子を使用して、Performance Monitor が使用する初期表示モードを指定できます。統計は、Histogram 修飾子を指定すると、最初はグラフ表示モードで表示されます。Nohistogram 修飾子を指定すると（デフォルト）、最初は数値表示モードで表示されます。

Compaq Tru64 UNIX

次に示す Compaq Tru64 UNIX のコマンドを使用すると、Performance Monitor は、最初はグラフ表示モードで mf_personnel データベースに関する統計を表示します。

```
$ rmu -show statistics -histogram mf_personnel
```

◆

Performance Monitor がグラフ表示モードで統計を表示しているときに、「N」を入力すると、数値表示モードに切り替えることができます。

2.2.6.2 数値表示形式

数値表示形式では、統計を列で表示します。この表には、最大レートと現行レートだけでなく、平均値と合計値が表示されます。次に、数値表示形式の例を示します。

```
Node: TRIXIE         Oracle Rdb V7.0-00 Performance Monitor 17-MAY-1996 14:37:03
Rate: 3.00 Seconds           Summary IO Statistics          Elapsed: 05:39:13.29
Page: 1 of 1          SQL$DISK1:[USER]MF_PERSONNEL.RDB;1            Mode: Online
------------------------------------------------------------------------------
statistic.........      rate.per.second.........     total...     average..
name..............      max.     cur.     avg...     count...     per.trans
transactions               0        0       0.0           2           1.0
verb successes             2        0       1.7         250         125.0
verb failures              0        0       0.2          32          16.0
synch data reads           0        0       0.4          66          33.0
synch data writes          0        0       0.0           0           0.0
asynch data reads          0        0       0.0           0           0.0
asynch data writes         0        0       0.0           0           0.0
RUJ file reads             0        0       0.0           0           0.0
RUJ file writes            0        0       0.0           0           0.0
AIJ file reads             0        0       0.0           0           0.0
AIJ file writes            0        0       0.0           0           0.0
ACE file reads             0        0       0.0           0           0.0
ACE file writes            0        0       0.0           0           0.0
root file reads            0        0       0.1          20          10.0
root file writes           0        0       0.0           5           2.5
------------------------------------------------------------------------------
Exit Graph Help Menu Options Pause Reset Set_rate Time_plot Unreset Write X_plot
```

次に、数値表示形式のレイアウトを説明します。

- statistic name
  この列は統計の名称を示します。表 2-5（2-47）に、各画面とその説明が記載されている場所を示します。

- rate per second (max)
  Performance Monitor が起動されて以降の、このイベントの 1 秒あたりの最大レートです。[R] を入力して「Reset」オプションを選択すると、このカウンタをリセットできます。
- rate per second (cur)
  直前のサンプル間隔中の、そのイベントの 1 秒間の発生レートを示します。RMU Show Statistics コマンドに Time 修飾子を指定すると、サンプル間隔をあらかじめ設定できます。また、[S] を入力して「Set_rate」オプションを選択すれば表示した後でも変更できます。
- rate per second (avg)
  データベースがこのノードでオープンされて以降の、このイベントの 1 秒あたりの平均発生レートを示します。このカウンタをリセットするには、[R] を入力して、「Reset」メニュー・オプションを選択します。カウンタをリセットした後、この列には、カウンタがリセットされて以降の 1 秒あたりのイベントの平均発生レートが表示されます。
- total count
  データベースがこのノードでオープンされて以降、このイベントが発生した総件数を示します。このカウンタをリセットするには、[R] を入力して、「Reset」メニュー・オプションを選択します。カウンタをリセットすると、この列は、カウンタがリセットされた以降に発生したイベントの総件数を示します。
- average per trans
  total-count 列を、完了したトランザクション数で割ったものです。

[No]Histogram 修飾子を使用すると、Performance Monitor が使用する初期表示モードを指定できます。統計は、Histogram 修飾子を指定すると、最初はグラフ表示モードで表示されます。Nohistogram 修飾子を指定すると（デフォルト）、最初は数値表示モードで表示されます。

Compaq Tru64 UNIX

次の Compaq Tru64 UNIX のコマンドを使用すると、Performance Monitor は、最初は数値表示モードで mf_personnel データベース用の統計を表示します。

```
$ rmu -show statistics -nohistogram mf_personnel ♦
```

Performance Monitor が数値表示モードで統計を表示している場合、[G] を入力すると、グラフ表示モードに切替えることができます。

2.2.6.3 タイム・プロット表示形式

タイム・プロット表示形式は、画面上の特定のフィールドに関するイベント件数の詳細なグラフを提供します。各画面には、水平メニューがあります。そのメニューに「Time_plot」があれば、[T] を入力してタイム・プロットを要求できます。[T] を入力した後、該当フィールドに関連する文字を入力して画面からフィールドを選択します（または矢印キーを使用してカーソルを該当フィールドに移動し、[Enter] キーを押します）。

次に、Verb の成功レートを示すタイム・プロットの例を示します。

```
Node: TRIXIE         Oracle Rdb V7.0-00 Performance Monitor 17-MAY-1996 14:41:24
Rate: 3.00 Seconds            Summary IO Statistics         Elapsed: 00:07:35.33
Page: 1 of 1           SQL$DISK1:[USER]MF_PERSONNEL.RDB;1           Mode: Online
-------------------------------------------------------------------------------
                                verb successes
Maximum rate = 147           Current rate = 144           Average rate = 78.0
Total count = 10216                            Average per transaction = 8.9
         +---------+---------+---------+---------+---------+---------+---------+
100+     |        ^^^^^^^^**^^^^         |         |         |         |         |
 89-99   |       *  |         |         |         |         |         |         |
 78-88   |  **      |         |         |         |         |         |         |
 67-77   |    **    |         |         |         |         |         |         |
 56-66   |*         |         |         |         |         |         |         |
 45-55   |   *      |         |         |         |         |         |         |
 34-44   |          |         |         |         |         |         |         |
 23-33   |          |         |         |         |         |         |         |
 12-22   |          |         |         |         |         |         |         |
  1-11   |          |         |         |         |         |         |         |
         +---------+---------+---------+---------+---------+---------+---------+
                        Sample interval is 3.00 seconds
-------------------------------------------------------------------------------
Exit Help Menu Normal Pause Reset Set_rate Unreset Write !
```

上の列のサーカムフレックス記号（ ^ ）は、イベント数（この場合は、成功した Verb の数）がチャートの最大値を超えていることを示しています。画面の範囲を変更するには、「Reset」メニュー・オプションを選択します。

「Set_rate」メニュー・オプションを使用してサンプル間隔を変更すると、この画面は新しい間隔に変更されます。たとえば、次の「Summary I/O Statistics」表示では 1 秒のサンプル間隔を使用しています。

```
Node: TRIXIE         Oracle Rdb V7.0-00 Performance Monitor 17-MAY-1996 14:42:44
Rate: 1.00 Seconds            Summary IO Statistics         Elapsed: 00:03:58.19
Page: 1 of 1          SQL_DISK1:[ORION]MF_PERSONNEL.RDB;1           Mode: Online
-------------------------------------------------------------------------------
                                verb successes
Maximum rate = 147           Current rate = 87            Average rate = 89.5
Total count = 21663                            Average per transaction = 9.0
         +---------+---------+---------+---------+---------+---------+---------+
154+     |         |         |         |         |         |         |         |
137-153  |      *  |         |         |         |         |         |         |
120-136  | *   *   |         |         |         |         |         |         |
103-119  |*   *  **|  *** ** *   *     |         |         |         |         |
 86-102  |         | *   *  *|*   * *   |         |         |         |         |
 69-85   |         |         |  *      |         |         |         |         |
 52-68   |  **     *         | *   *   |         |         |         |         |
 35-51   |         |*        |         |         |         |         |         |
 18-34   |         |         |         |         |         |         |         |
  1-17   |         |         |         |         |         |         |         |
         +---------+---------+---------+---------+---------+---------+---------+
                          Sample interval is 1.00 seconds
-------------------------------------------------------------------------------
Exit Help Menu Normal Pause Reset Set_rate Unreset Write !
```

ノートパッド機能は、Time_plot のベースラインを有効または無効にできるオプションを提供しています。このベースラインを有効にしておくと、グラフを継続的に更新します。次のステップを実行すると、Time_plot ベースラインを有効にできます。

1. 感嘆符（[!]）を入力して、ツール機能を起動します。
2. 「Select Tool」サブメニューから「Notepad」オプションを選択します。
3. 「Select Tool Option」サブメニューから「Enable Time_plot」オプションを選択します。

2.2.6.4 散布図プロット表示形式

散布図プロット表示形式は、タイム・プロット表示形式と類似しており、現在の画面上に選択した統計フィールドの情報を表示します。しかし、散布図プロット内の情報は垂直ヒストグラムで表示されます。散布図プロット表示形式を使用すると、統計の平均レートを決める値の構成を決めることができます。メニューに「X_plot」があれば、[X] を入力して散布図プロットを要求できます。[X] を入力した後、該当フィールドに関連する文字を入力して画面からフィールドを選択します（または矢印キーを使用してカーソルを該当フィールドに移動し、[Enter] キーを押します）。

このフィールドを選択した後では、散布図プロット表示は、当初は選択したフィールドの継続的な平均レートに基づくスケールで表示されます。このスケールは「Reset」メニュー・オプションを使用して変更できます。

画面の各縦線は、選択した統計レートに基づく収集レートのバケットを意味しています。バケットの全体集合は収集範囲と呼ばれ、散布図画面全体を構成します。たとえば、散布図画面が 0.0 ～ 1.5 秒の範囲であれば、各バケットは 1 秒の 0.1（10 分の 1）を表しています。

選択した画面のリフレッシュ・レートに基づいて、選択した統計フィールドに対する正規化された現行のレートを検査して、そのレートがどのバケットに入るかを決定します。このアルゴリズムによって、現行レートは分散グラフの分布で表示され、これはパフォーマンス分析にきわめて便利です。

次の例では、現行レートの分散が継続的な平均（最下行の水平軸上の文字「R」で示される）よりもきわめて大きいことから、興味深い散布図プロット表示を示しています。これは、最近、非同期データ読込みアクティビティが急に増加するイベントが発生したことを示しています。また、データの分散がゆがんでいることにも注意してください。

```
Node: ORANOD          Oracle Rdb V7.0-00 Performance Monitor 29-MAY-1995 10:13:33
Rate: 0.50 Seconds            Summary IO Statistics        Elapsed: 2 19:36:03.15
Page: 1 of 1          CADD$:[J_DOE.WORK.ALS]MF_PERSONNEL.RDB;1       Mode: Online
--------------------------------------------------------------------------------
Out-of-range count:        0.1--- = 0             (1) (2-32) 3.0+++ = 29
Current statistic range:      low = 0.8           (2) (2-33)   high = 4.0
                              synch data writes   (3) (2-33)
Scaled distribution of statistic rate (in seconds)(4) (2-33)       Elapsed:
00:25:00.51
+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+---+
|    |    |    |    |    |    |    |    |    ***  |    **** |    |    |    |    |   |
|    |    |    |    |    |    |    |    |    ******    *****|    |    |    |    |   |
|    |    |    |    |    |    |    |    |    ***************|    |    |    |    |   |
|    |    |    |    |    |    |    |    |    ******************  |    |    |    |   |
|    |    |    |    |    |    |    |    |    ********************|    |    |    |   |
|    |    |    |    |    |    |    |    |    ********************|    |    |    |   |
|    |    |    |    |    |    |    |    |    *********************    |    |    |   |
|    |    |    |    |    |    |    |    |    *************************|    |    |   |
|    |    |    |    |    |    |    |    |    ***************************** |    |   |
|    |    |    |    *    *    ***  ****  ********************************** ***  |
+----+----+----+----+----+-R--+----+----+----+----A----+----+--M-+----+----+---+
                          (5) (2-33)                  (6) (2-33)         (7) (2-33)
0.0  0.2  0.4  0.6  0.8  1.0  1.2  1.4  1.6  1.8  2.0  2.2  2.4  2.6  2.8  3.0+
                  (Each "*" represents 11 statistics updates)
--------------------------------------------------------------------------------
Config Exit Help Menu Normal Pause Reset Set_rate Write !
```

この例は、表示のサマリー領域について次の種類の情報を示しています。

(1) 記録される最小レートよりも低い（XXX---）か、最大レートよりも高い（XXX+++）統計更新の回数を示します。これは、境界データ条件についての情報です。

(2) 記録する最低レートおよび最高レートを識別します。これは、画面上の「Reset」メニュー・オプションを選択したときに、画面のスケーリングの再設定基準を決めるために使用されます。これは、境界データ条件についての情報でもあります。

(3) リージョンは、レートの分散情報の収集対象となる統計を示します。

(4) サマリー領域の 4 行目は、画面のスケール変更係数と収集にかかった総経過時間を示します。

3 つのスライディング・インジケータが、最下部の水平軸に沿って表示されます。

(5) 文字「R」は、継続的な平均レートを示しています。これは、サマリー領域に示されているように、総経過時間を超えて計算されています。継続的な平均レート「R」はいつも表示されるわけではなく、収集の範囲によることに注意してください。また、経過時間のリセットは、収集した平均値と継続的な平均値を比較する場合に便利です。

(6) 文字「A」は、収集された平均レートであり、収集時間に基づいて計算された値を示しています。

(7) 文字「M」は、収集されたレートの中央値であり、収集時間に基づいて計算された値を示しています。

収集バケットの数は、端末の表示幅を基準にしています。デフォルトでは、各バケットは 5 列で構成されています。列の数は、ユーザーが指定できます。たとえば、端末の幅を 132 列に設定すると、端末の幅が 80 列の場合よりも多くの情報を表示できます。バケットあたりの列の数を減らすと、80 列の端末幅でも同じ効果が得られますが、精度は失われます。

収集された中央値は、収集された平均値よりもきわめて大きいことに注意してください。これはゆがんだデータを示しており、一般的には、実行時のパフォーマンスの周期的なブリップによって発生します。これらのブリップを識別し、除去すると、よりスムーズな実行システムが得られます。

随時発生するこれらのブリップは、実行時の平均からノイズとして取り除かれる傾向にあることにも注意してください。しかし、これらのブリップを取り除くことで、システムの継続的な平均がきわめて大きく改善されます。

理想としては、散布図プロット表示は、伝統的な標準分布曲線のデータ分布に類似している必要があります。次の例では、この状態を示します。

```
Node: ORANOD         Oracle Rdb V7.0-00 Performance Monitor 28-MAY-1995 21:27:09
Rate: 0.50 Seconds           Summary IO Statistics        Elapsed: 2 06:49:39.77
Page: 1 of 1         CADD$:[J_DOE.WORK.ALS]MF_PERSONNEL.RDB;1       Mode: Online
--------------------------------------------------------------------------------
Out-of-range count:        12.7--- = 3              35.2+++ = 12
Current statistic range:       low = 10.6              high = 38.0
                               asynch data reads
Scaled distribution of statistic rate (in seconds)          Elapsed: 00:45:12.80
+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+---+
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |
|    |    |    |    |    |    |    |    |    **** |    |    |    |    |    |   |
|    |    |    |    |    |    |    |    |    ********* |    |    |    |    |   |
|    |    |    |    |    |    |    |    |    **********|    |    |    |    |   |
|    |    |    |    |    |    |    |    |    **********|    |    |    |    |   |
|    |    |    |    |    |    |    |    |    **********|    |    |    |    |   |
|    |    |    |    |    |    |    |    |    **********|    |    |    |    |   |
|    |    |    |    |    |    |    |    ***  ************   *    ***  |    |   |
|    |    |    |    |    *    *    ***  ******************************|    |   |
*    *    *    **** ************************************************************ |
+----+----+----+R---+----+----+----+----+-M--+----+A---+----+----+----+----+---+
11.2 12.8 14.4 16.0 17.6 19.2 20.8 22.4 24.0 25.6 27.2 28.8 30.4 32.0 33.6 35.2
                     (Each "*" represents 4 statistics updates)
--------------------------------------------------------------------------------
Config Exit Help Menu Normal Pause Reset Set_rate Write !
```

散布図プロット画面には、次のメニュー・オプションが含まれています。

- Normal
  通常の画面表示に戻ります。散布図プロット表示に入る前に表示されていた画面が表示されます。
- Reset
  統計をリセットするかわりに、実際に記録された最高および最低のレートを基準にして散布図プロット・スケールをリセットします。このメニュー・オプションは、実際に発生した統計の更新レートに注目する場合に非常に便利です。
- Set_rate
  散布図プロット情報をクリアして、データの収集を再開します。これは、新しいレートのデータ収集パターンと古いレートのデータ収集パターンがまったく関係がない場合に必要です。
- Config
  Config を使用すると、レート収集パラメータと表示の外観を微調整できます。現在、バケットごとに表示される列数の設定、およびレート収集範囲の設定の、2 通りの設定操作が行えます。この 2 つの設定方法を使用することにより、どのデータを収集して表示するか、さらに精密に制御できるようになります。

バケットごとの列数を減少させると、より多くのレートを収集できます。逆に、バケットごとの列数を増加させると収集するレートは減少しますが、バケットごとのレートの報告はより正確になります（スケーリングが向上）。最小の列数は 2 で、最大の列数は 10 です。

特定の収集範囲の値を設定することで、通常はすでに収集したレートに基づいて、実際の収集範囲を決定できます。

散布図プロット画面情報は、Output 修飾子を使用して作成したバイナリ出力ファイルに記録され、Input 修飾子を使用して再生できます。記録フェーズ中に収集レートを変更すると、再生中に識別されないため、散布図プロット表示が奇異な結果になることがあるので注意してください。

2.2.6.5 テーブル表示形式

画面によっては、テーブル表示形式でのみ利用可能なものもあります。たとえば、「Stall Messages」、「Active User Stall Messages」および「Defined Logicals」表示は、テーブル表示形式で表示します。

次に「Defined Logicals」表示の例を示します。

```
Node: TRIXIE          Oracle Rdb V7.0-00 Performance Monitor 17-MAY-1996 14:44:03
Page: 1 of 3                     Defined Logicals            Elapsed: 05:46:46.72
Rate: 3.00 seconds     SQL$DISK1:[USER]MF_PERSONNEL.RDB;1             Mode: Online
--------------------------------------------------------------------------------
Logical.Name................. Table.Name......... Logical.Definition.........
RDM$BIND_BUFFERS              LNM$SYSTEM_TABLE    80
SQL$DATABASE                  LNM$PROCESS_TABLE   SQL_PERSONNEL
RDM$MAILBOX_CHANNEL           LNM$SYSTEM_TABLE    MBA2563:
RDM$MONITOR                   LNM$SYSTEM_TABLE    SYS$SYSROOT:[SYSEXE]
--------------------------------------------------------------------------------
Exit Full Help Menu >next_page <prev_page Set_rate Write !
```

2.2.7 文字セル・インタフェースにおける「Zoom」メニュー・オプションの使用方法

ズーム画面は、特定の画面項目、たとえば一般的にはロック、プロセス、AIJ、記憶領域についての詳細な情報を表示するサブウィンドウです。ズーム画面は、情報の静的なスナップショットであることに注意してください。一般的な統計画面とは異なり、この情報は変化しません。ズーム機能を持つ画面では、水平メニューにズーム・オプションが表示されます。

記憶領域やアクティブ・データベース・プロセスの詳細な情報を迅速に表示することが必要な場合が多くあります。一般に、この要求は、領域ごとまたはプロセスごとに画面を表示している場合で、さらに調査を保証するイベントが発生した場合に発生します。

たとえば、次の「Checkpoint Information」画面では、プロセスの 1 つがチェックポイントの対象でないことに注目してください。

```
Node: ORANOD         Oracle Rdb V7.0-00 Performance Monitor 21-APR-1996 06:30:30
Rate: 0.10 Seconds           Checkpoint Information         Elapsed: 00:28:45.14
Page: 1 of 1        DD$:[ANDERS.WORK.ALS]MF_PERSONNEL.RDB;12        Mode: Online
--------------------------------------------------------------------------------
Process.ID  Ckpt.Vno:Ckpt.Vbn  QuietVno
2020D41F:1s       32      106         0
2020EC26:1s                           0
20208957:1        32      132         0
20207EF7:1        32      119         0
202084F9:1        32      135         0
--------------------------------------------------------------------------------
Config Exit Help Menu >next_page <prev_page Pause Set_rate Write Zoom !
```

「Zoom」メニュー・オプションを選択すると、なぜプロセス 2020EC26 はチェックポイントの対象ではないかという情報が表示されます。[Z] キーを押すと、現在表示されているプロセス ID のメニューが表示されるので、該当するプロセス ID を選択します。Performance Monitor は、指定したプロセスに対する詳細情報を表示します。次の表示とよく似たズーム画面が表示されます。

```
Node: ORANOD         Oracle Rdb V7.0-00 Performance Monitor 21-APR-1996 06:32:11
Rate: 0.10 Seconds           Checkpoint Information         Elapsed: 00:30:26.38
Page: 1 of 1     DD$:[ANDERS.WORK.ALS]MF_PERSONNEL.RDB;12           Mode: Online
--------------------------------------------------------------------------------
Process.ID  Ckpt.Vno:Ckpt.Vbn  QuietVno
2020D41F:1s       34       89         0
2020EC26:1s                           0
20208957:1        34      164         0
20207EF7:1        34      169         0
202084F9:1        34      163         0
+--Process Information: 2020EC26---------------------------------+
|                                                                 |
| Active user with process ID 2020EC26 (database server)          |
| User name is RDBVMS                                             |
| Process name is RDM_LCS701_1                                    |
| Image name is DSA1:[SYS1.SYSCOMMON.][SYSEXE]RDMLCS701.EXE;206   |
| No transaction in progress                                      |
| Monitor ID is 1                                                 |
| Internal Stream ID is 1                                         |
| Internal Transaction ID is 655                                  |
|                                                                 |
+-----------------------------------------------------------------+
--------------------------------------------------------------------------------
Exit Help Menu >Next_page <Prev_page Pause Set_rate Write Zoom !
```

このように、プロセス 2020EC26 は、データベース・サーバー・プロセスであり、実行中のトランザクションは含みません。

2.2.8 文字セル・インタフェースでの画面のファイルへの書込み

Performance Monitor のメイン・メニューから「Options」を選択すると、STATISTICS.RPT という名前のファイルに、特定時点のほとんどの画面を書き込むことができます。ファイルは、図式または数値、またはその両方のフォーマットで書き込むことができます。

メニューから「Options」を選択すると、Oracle Rdb では、次のような画面が表示されます。

```
+---Select Report Format---+
|                          |
|  Write Report (Graph)    |
|  Write Report (Numbers)  |
|  Write Report (Both)     |
|                          |
+--------------------------+
```

次の機能を選択できます。

- Write Report (Graph)
  このコマンドを実行すると、すべての「Graph」画面が STATISTICS.RPT という名前のファイルに書き込まれます。
- Write Report (Numbers)
  このコマンドを実行すると、すべての「Numbers」画面が STATISTICS.RPT という名前のファイルに書き込まれます。
- Write Report (Both)
  このコマンドを実行すると、「Graph」画面と「Numbers」画面の両方が STATISTICS.RPT という名前のファイルに書き込まれます。

「Write」オプションを選択すると、「Help」を除くあらゆる画面の内容を書き込むことができます。このオプションを選択すると、デフォルト・ディレクトリに RMU.SCR という名前のファイルが作成されます。

2.2.9 文字セル・インタフェースでの Performance Monitor ツールの使用

Performance Monitor は、ノートパッドの作成とメンテナンスを行い、事前定義およびユーザー定義のコマンドを起動するツール群を提供しています。コマンドは、Performance Monitor の実行中に起動できます。

任意の画面やメニューで感嘆符 [!] を入力すると、ツール機能を起動できます。たとえば、「Select Tool」サブメニューが次のように表示されます。

```
+------ Select Tool --------+
|                           |
|  A. Goto screen "by-name" |
|  B. Goto screen       [-> |
|  C. Notepad (EDT)         |
|  D. Edit a file (EDT)     |
|  E. Send/Receive Mail     |
|  F. Invoke command        |
|  G. Wake up process       |
|  H. Terminate image       |
|  I. Terminate process     |
|  J. Set process priority  |
|  K. Switch database       |
|  L. Start screen cycling  |
|  M. Start stall logging   |
|  N. Start DBKEY logging   |
|  O. Disable auto-refresh  |
|  P. Start ALS process     |
|  Q. <<more>>              |
+---------------------------+
```

オプション「Q」を選択すると、次の画面が表示されます。

```
+------ Select Tool --------+
|                           |
|  A. <<more>>              |
|  B. Wake up RCS process   |
|  C. Invoke ABS process    |
|  D. Suspend ABS process   |
|  E. Notepad options   [-> |
+---------------------------+
```

「Select Tool」サブメニューは、表 2-4（2-38）のオプションをサポートします。

表 2-4 「Select Tool」オプション

オプション	説明
Goto screen "by-name"	移動先の画面名を、大文字 / 小文字を区別しないで入力します。また、画面名を構成する文字列の一部でも入力できます。画面名の一部を入力する場合は、十分な長さを入力しないと画面名が一意に選択されないので注意してください。一意に判断できない画面名を入力すると、最初に一致した画面が選択されます。選択の順序はアルファベット順とは限らないことに、注意してください。一致する画面名が存在しない場合、Pefromance Monitor はエラーを表示し、現行画面が表示されたままになります。

表 2-4 「Select Tool」オプション（続き）

オプション	説明
Goto screen [->	使用可能なすべての画面名を、アルファベット順に複数ページのメニューで表示します。Peformance Monitor で使用可能かどうかによって、選択できる画面名が決まります。たとえば、グローバルなバッファ画面名は、実際に使用可能な状態でなければ選択できません。
Notepad(EDT)[*1]	ノートパッドは、セッションに関するメモを記録できるファイルです。ノートパッド・ファイルの名前は、「Notepad」オプションのサブメニューで指定できます。デフォルトのノートパッド・ファイル名は、STATS.NOTEPAD です。 ノートパッド・ファイルは、EDT、LSE、TPU の 3 種類のテキスト・エディタで編集できます。エディタを変更するには、「Options」サブメニューを選択するか、RMU$EDIT 論理名を定義します。RMU$EDIT 論理名の値は、EDT、LSE または TPU でなければなりません。デフォルトのテキスト・エディタは、EDT です。EDTINI.EDT ファイルが存在する場合、テキスト・エディタは、起動時にそのファイルを読み込みます。 ノートパッド・ファイルが存在しない場合、指定のテキスト・エディタが起動する前に、4 行の情報を格納したファイルが自動的に作成されます。この 4 行には、製品のバージョン、データベース名、ノートパッド・ファイルの名前、ファイルが作成された日付と時間が含まれます。 ノートパッド・ファイルの名前には、画面の書込みに使用するファイル名である RMU.SCR のように、いずれかの Performance Monitor ファイルの名前を定義すると便利です。これで、「Write」メニュー・オプションを使用して特定の画面イメージを作成し、次にノートパッド機能を使用してドキュメントを作成できます。
Edit a file (EDT)[*1]	テキスト・エディタを起動します。「Filename:」のプロンプトが表示されるので、オプションでファイル名を入力できます。ファイル名を指定したくない場合は、[Enter] キーを押してください。 テキスト・エディタは、EDT、LSE、TPU の 1 つを起動できます。エディタを変更するには、「Options」サブメニュー（後で説明）を選択するか、RMU$EDIT 論理名を定義します。RMU$EDIT 論理名の値は、EDT、LSE または TPU でなければなりません。デフォルトのテキスト・エディタは、EDT です。EDTINI.EDT ファイルが存在する場合、テキスト・エディタは、起動時にそのファイルを読み込みます。
Send/Receive Mail[*1]	メール・ユーティリティを起動します。

表 2-4 「Select Tool」オプション（続き）

オプション	説明
Invoke command	ユーザー定義のコマンドを起動します。OpenVMS では、現行の画面の最下行に DCL ($) プロンプトが表示されるので、DCL コマンドを入力できます。 また、OpenVMS では、ドル記号（[$] キー）を入力することで、どの画面からでも DCL コマンドを起動できます。現在の画面の最下行にプロンプト「$」が表示されるので、任意の DCL コマンドを入力してください。 Compaq Tru64 UNIX では、現在の画面の最下行に山カッコ（>）が表示されるので、任意の Compaq Tru64 UNIX コマンドを入力できます。 データベースを操作するユーティリティ（SQL など）を起動する場合は、注意が必要です。排他的アクセスが必要なコマンドを起動すると、ハングした状態になることがあります。
Wake up process[*2]	休止状態のデータベース・プロセスを起動します。このオプションが必要になるのは、非常に重大なデータベース障害が発生し、データベース・リカバリ・プロセス（database recovery process: DBR）を実行してもプロセスを起動できなかった場合のみです。このオプションは、休止状態以外のプロセスには影響しませんが、過度に使用しないよう注意してください。このオプションは、特定のデータベース・プロセスでの操作に限定されているため、システム上の任意のプロセスに対して使用することはできません。
Terminate image[*2]	"%RDMS-F-IMGABORTED, image aborted at privileged user request." という例外を除いて、データベース・イメージを終了します。終了するのはイメージのみで、プロセス自体は終了しないことに注意してください。この操作では、プロセスのリカバリ中に、データベースがフリーズしないので便利です。このオプションは、特定のデータベース・プロセスでの操作に限定されているため、システム上の任意のプロセスに対して使用することはできません。
Terminate process[*2]	警告なしでデータベース・プロセスを終了します。この操作を行うと、終了したプロセスを DBR で復旧している間、データベースはフリーズします。このオプションは、特定のデータベース・プロセスでの操作に限定されているため、システム上の任意のプロセスに対して使用することはできません。
Set process priority[*1]	OpenVMS 上の指定したプロセスの優先順位を変更できます。指定したプロセスの基本プロパティよりも高い優先順位を設定するには、ALTPRI（優先順位変更）権限が必要です。

表 2-4 「Select Tool」オプション（続き）

オプション	説明
Switch database	Performance Monitor を終了せずに、データベースの切替えができます。オープンしたい新規データベースのファイルをプロンプトで指定します。コマンド・リコールを実行すると、それ以前の端末入力を表示できます。リクエストをキャンセルするには、OpenVMS では [Ctrl] キーを押しながら [Z]、Compaq Tru64 UNIX では [Ctrl] キーを押しながら [D] を押します。 新規データベースのファイル指定を入力したら、現在のデータベースはクローズされ、新しいデータベースがオープンします。なんらかの理由で新しいデータベースをオープンできない場合は、前にオープンしていたデータベースが再度オープンします。INPUT 修飾子を使用したバイナリ入力ファイルの再生や、OUTPUT 修飾子を使用したバイナリ出力ファイルの記録を実行している間は、新しいデータベースをオープンできません。
Start screen cycling	現在選択している画面に対応付けられた複数の画面を、順番に Performance Monitor の対象にします。各画面は、指定された秒数の間表示されます。 Performance Monitor による画面の巡回では、画面モードの変更やサブメニューの変更ができます。選択内容に関連付けられている画面の巡回は、現在選択しているメニュー・レベルで継続します。 「Start screen cycling」オプションを選択すると、プロンプトが表示されるので間隔（秒）を入力します。間隔には、TIME 修飾子に指定した値以上の値を指定してください。さらに、リフレッシュ・レートを、間隔で指定した値よりも大きな値に手動で変更（「Set_rate」メニュー・オプションを使用）すると、「Set_rate」メニュー・オプションで指定した間隔で巡回します。
Start stall logging	ストール・メッセージをストール・ログ・ファイルに書き込むよう指定します。プロンプトに対して、ストール・ログ・ファイルのファイル名を入力してください。大量のストール・メッセージが生成されているのに気づいた場合で、問題を迅速に調査、解決するリソースがない場合、ストール・メッセージをログ・ファイルへ書き込んでおくと便利です。生成したログ・ファイルを後で参照し、問題のトレースや解決に使用できます。

表 2-4 「Select Tool」オプション（続き）

オプション	説明
	ストール・メッセージは、「Stall Messages」画面に類似したフォーマットで書き込まれます。ストール・メッセージは、画面のリフレッシュ・レートで書き込まれます。リフレッシュ・レートに大きな値を指定すると、ファイルのサイズは小さくなりますが、書き込まれないストール・メッセージが多くなります。リフレッシュ・レートに小さな値を指定すると、ファイルのサイズは大きくなりますが、ほとんどのストール・メッセージが書き込まれます。 ストール・メッセージをログ・ファイルに記録するのに、「Stall Messages」画面を表示しておく必要はありません。表示されている画面に関係なく、ストール・ログへの記録は実行されます。 デフォルトでは、ストール・メッセージはログ・ファイルに書き込まれません。
Start DBKEY logging	所定の処理時間内に、様々な接続プロセスがアクセスしたレコードをログに記録します。プロンプトに対して、アクセスした dbkey を記録するファイル名を入力してください。 dbkey 情報は、現在の画面のリフレッシュ・レートで書き込まれます。リフレッシュ・レートに大きな値を指定すると、ファイルのサイズは小さくなりますが、書き込まれない dbkey メッセージが多くなります。リフレッシュ・レートに小さな値を指定すると、ファイルのサイズは大きくなりますが、ほとんどの dbkey メッセージが書き込まれます。 dbkey メッセージをログ・ファイルに記録するのに、「DBKEY Information」画面を表示しておく必要はありません。表示されている画面に関係なく、dbkey ログの記録は実行されます。
Disable auto-refresh	Performance Monitor のメニューをブロードキャスト・メッセージで上書きします。デフォルトでは、メッセージの表示後に水平メニューをリフレッシュします。このオプションを指定すると、画面のリフレッシュを無効にできます。
Start ALS process	AIJ が有効化されている場合、AIJ ログ・サーバー・プロセスを起動または停止します。
Wake up RCS process	行キャッシュ・サーバー・プロセスに対して、しきい値に達していなくてもキャッシュの破棄を開始するよう指示します。
Invoke ABS process	現行以外の指定の AIJ に対して、AIJ バックアップ・サーバー（AIJ Backup Server：ABS）プロセスを起動します。
Suspend ABS process	ABS プロセスを一時停止または再開します。

表 2-4 「Select Tool」オプション（続き）

オプション	説明
Notepad options [->[*3]	様々なツール・オプションを変更します。「Select Tool Option」サブメニューが表示されます。 オプション「A」は、EDT テキスト・エディタを選択します。これは、デフォルトのテキスト・エディタです。 オプション「B」は、LSE テキスト・エディタが存在すれば、これを選択します。LSE エディタが存在しない場合は、EDT エディタが選択されます。 オプション「C」は、TPU テキスト・エディタが存在すれば、これを選択します。TPU エディタが存在しない場合は、EDT エディタが選択されます。 オプション「D」は、ノートパッド・ファイル名を変更します。ノートパッド・ファイル名のデフォルトは、STATS.NOTEPAD です。 オプション「E」は、Time_plot ベースラインを有効または無効にします。ベースラインを有効にすると、Time_plot ディスプレイ・グラフを常に更新します。 「Select Tools Options」サブメニューを終了するには、OpenVMS では [Ctrl] キーを押しながら [Z]、Compaq Tru64 UNIX では [Ctrl] キーを押しながら [D] を押します。

[*1] OpenVMS システムのみでサポートされます。
[*2] OpenVMS では、同じグループ内の別のプロセスを操作するためには、プロセスのユーザー識別コード（User Identification Code：UID）が起動プロセスと同一である場合を除いて、Group 権限が必要です。システム内の他のプロセスを操作するには、World 権限が必要です。
[*3] Compaq Tru64 UNIX でサポートされる「Notepad」オプションは、Time_plot ベースラインの有効化のみです。

2.2.10 文字セル・インタフェースでのオンライン・ヘルプ

幅広いオンライン・ヘルプ機能が提供されているので、統計データの理解と解析の参考にしてください。メニューの [H] キーまたはヘルプ・キーである [PF2] キーを押すと、Performance Monitor の「Help」メニューを選択できます。次に、Performance Monitor の「Help」メニューの例を示します。

```
A. Using the keyboard
B. Screen description
C. Field descriptions    [->
D. Search for help on... [->
E. Goto screen "by-name"
F. Goto screen           [->
```

「Help」メニューには、使用可能なヘルプ・トピックが表示されます。「Using the keyboard」を選択すると、メニューの選択肢やその他のキーについての説明がヘルプ・テキストに表示されます。「Screen description」を選択すると、現在の画面の説明が表示されます。「Field descriptions」を選択するときには、ヘルプを表示したいフィールドに関連した文字を押すか、矢印キーを使ってフィールドの上にカーソルを置きます。次に、[Enter] キーを押すと、フィールドの説明が表示されます。フィールドのヘルプ・テキストがない場合、「Help」メニューに「Field descriptions」は表示されません。

キーボード、画面、フィールドなどのヘルプ・テキストは、複数のページにまたがることがあります。ヘルプ画面が複数ある場合は、一度に 1 行または 1 ページずつスクロールできます。[↑] キーを押すと上方向に 1 行ずつ、[↓] キーを押すと下方向に 1 行ずつ移動できます。[Page Up] または [←] キーを押すと 1 ページずつ戻り、[Page Down] または [→] キーを押すと 1 ページずつ進みます。

前後方向に 1 行以上のヘルプ・テキストがある場合に、矢印キー、[Page Up] キーおよび [Page Down] キーがアクティブになります。キーがアクティブになっているかどうかは、画面の最下行に表示されます。キーがアクティブではない状態で [←]、[→]、[Page Up]、[Page Down] などのキーを押しても、何も実行されません。また、1 行以上のヘルプ・テキストが後に存在する場合は "MORE-->"、1 行以上のヘルプ・テキストが前に存在する場合は "<--MORE" が表示されます。

ヘルプ画面の一番上には、現在表示されているヘルプ・トピックの名前が表示されます。

ヘルプ・テキストのサイズは、端末に設定された行数が基準になります。たとえば、48 行の DECwindows 端末の表示領域は、24 行の VT100 端末よりも大きくなります。

2.2.11 データベース統計の種類

Performance Monitor で表示できる統計は、次の 4 つのカテゴリに分類されます。

- トランザクションのアクセス・パターンやシステム上の負荷など、データベースで発生している処理に関する一般情報を提供する統計。
- パラメータに関連する統計。パラメータの設定を変更することによってデータベース・パフォーマンスを向上させるもの。たとえば、バッファ・プールがオーバーフローし、マークしたページをデータベースに書き戻す必要がある場合の PIO 統計など。
- バッファ情報や行キャッシュ情報など、データベース・パラメータの変更を反映して自動変更する、動的な情報を表す統計。
- データベースのオンライン分析が可能な統計。たとえば、「Transaction Analysis」画面では、Verb の成功と失敗の比率を表示。

データベース構造や使用方法によって、統計の解釈は異なります。データベースは最適な設計だという確信がある場合は、安定状態で統計分析を行います。この方法では、定期的に統計を監視することにより、通常のデータベース・パフォーマンスをよく把握してください。次に、基準値から外れたものはないかを調べ、データベース設計またはシステム・パラメータ（発生している問題によって判断）を調整して問題を解決します。データベース設計に確信がない場合で、パフォーマンスに問題が発生している場合は、統計を使って問題が発生している部分を特定します。

代表的な方法として、データベースのサイズが比較的小さい間に、アプリケーションを記述しておきます。データベースが小さい間は、許容できる程度のパフォーマンスを発揮できます。データベースが大きくなるに伴って、パフォーマンスが低下することがよくあります。このような場合には、データベースの成長に従って統計を収集し、トランザクションの割合の変化を比較してください。統計から、時間の経過に伴ってデータベースがどのように変化したかがわかります。統計を比較することにより、問題の原因を究明できます。様々なデータベース・サイズで収集した統計を比較することにより、どのレベルでパフォーマンスが低下し始めたかがわかります。

データベースをどのような方法で追跡する場合でも、次の統計は重要です。

- I/O 画面：
  Verb Successes 統計と Verb Failures 統計を比較します。成功率に比べて失敗率が高いと、多くの処理が失敗していることを示します。デッドロックが多すぎることが原因として考えられますが、アプリケーションによって異なります。
  .ruj File Reads と .ruj File Writes の統計を比較してください。.ru File Reads の値が大きすぎる場合は、過度にロールバックが発生していることを示します。.ruj File Writes が多すぎる場合は、競合が発生しているか、大量のデータを変更するトランザクションがあることを示します。1 つのトランザクションについて複数の書込みが発生するということは、実際にコミットする前に、マークしたページをデータベースに書き戻していることを示します（原因は、他のリカバリ・ユニットがページを要求、バッファ・プールのオーバーフロー、多数の更新を実行する非常に大きなトランザクションなど）。

- ロックの概要画面：
  Rqsts Not Queued、Rqsts Stalled、Rqst Deadlocked などの統計を調べてください。この統計の値が大きい場合（データベースの正常稼動時を基準）、競合が発生していることを示します。データベース設計が最適ではない場合があります。また、非同期システム・トラップ（Asynchronous Systen Traps：AST）のブロックの値が大きいか、アクセスが頻繁なデータでユーザー間に競合が発生している可能性があります。

- AIJ 画面：
  Records Written 統計と Blocks Written 統計を比較します。
  Records Written と Blocks Written の割合は、AIJ レコードのサイズ（つまり、変更されたレコードのサイズ）を示します。この統計は、参考値です。

- PIO 画面：
  Data Page Request 統計と SPAM Page Request 統計を比較します。SPAM フェッチ率がデータ・フェッチ率に比べて高い場合は、データ・アクセス・パターンを正確に反映しているだけである可能性があります。ただし、競合が発生している可能性もあるので、SPAM フェッチの値が大きい原因を調べてください。
  Pool Overflow 統計を調べます。各トランザクションのプール・オーバーフローが多く発生している場合は、バッファ・サイズや各ユーザーに割り当てるバッファ数を調整してください。また、グローバルなバッファを有効にすることも可能です。バッファの管理は、4.1.2 項（4-19）を参照してください。
  Unmark Buffer 統計を調べ、Pool Overflow と Lock Conflict 統計をチェックします。プール・オーバーフローやロック競合が原因でバッファがマークされない場合は、Oracle Rdb に過大な負荷がかかっています。ロック競合、またはバッファ・プールが小さすぎることが原因となっている場合もあります。
  Data File Reads 統計と Data File Writes 統計を比較してください。一般的に、この割合から、現在のトランザクション・ミックスの状態がわかります。

- スナップショット画面：
  Retrieved Records、Fetched Line、Read Snap Page などの統計を調べます。フェッチ行の値が取得レコードの値よりも大幅に大きい場合、読取り専用のトランザクションによるレコード・アクセスが大量に発生していることを示します。アクセス・パスが最適でないことが原因となる可能性もあります。
  また、Read Snap Page 統計は、データベース処理の量を示します。Read Snap Page の値が取得レコードの値よりも大きい場合、頻繁に更新されているレコードがフェッチされている可能性があります。この値が 0 ならば、更新トランザクションが変更したデータには、読取り専用のトランザクションはほとんどアクセスしないことを示します。
  Page Too Full 統計は、読取り / 書込みトランザクションに適用されます。この統計は、頻繁にアクセスされ、スナップショット領域を構成しているデータが、一部のデータ・ページに含まれていることを示します。
  Page In Use または Page Conflict 統計を調べてください。この統計は、スナップショット・ファイル内での競合を示します。スナップショット・ファイルの拡張が必要になっている場合もあります。大きな負荷を処理することによって競合を解消できるため、

ファイルは自動的に拡張されません。しかし、競合を緩和するために、スナップショット・ファイルを手動で拡張することも可能です。

2.2.12 Performance Monitor の画面

画面を移動するには、「Menu」オプションを選択すると、メイン・メニューが表示されます。次に、目的の画面を選択してください。

同時にアクティブにできる画面は 1 つのみですが、すべての統計カウンタ（最大値と平均値を含む）を監視できます。OUTPUT 修飾子を指定すると、バイナリの統計出力ファイルのフォーマットを変えずに、様々な画面を選択できます。ファイルには、現在の画面だけでなく、すべてのデータベース統計が含まれています。

表 2-5（2-47）は、各画面の簡単な説明と、詳細な内容の参照先を示しています。

表 2-5 Performance Monitor の各画面

画面	説明	参照先
Summary IO Statistics	データベース I/O 操作の概要	3.2.1.1 項（3-8）
Summary Locking Statistics	ロック操作の概要	3.8.1.3 項（3-56） 8.4 項（8-43）
Summary Object Statistics	すべてのデータベース・ルート・ファイル・オブジェクトの累計情報	Performance Monitor のヘルプ
Summary Cache Statistics	データベース内のすべての行キャッシュに関する累計情報	Performance Monitor のヘルプ
Summary Cache Unmark Statistics	行キャッシュの "unmark" 累計統計であり、行キャッシュの行をディスクに書き戻す処理の状態を示す。	Performance Monitor のヘルプ
Record Statistics	データ行アクティビティの概要	4.1.1.7 項（4-13）
Transaction Duration	リアルタイムのトランザクション処理に関する全体的なパフォーマンス	3.2.1.6 項（3-18）
Custom Statistics	ベースとなる統計情報をカスタマイズした内容を表示（領域ごとでない統計やプロセスごとでない統計など）	2.2.14 項（2-54）
Snapshot Statistics	更新と読取り専用のトランザクションでの、スナップショット処理	4.1.1.10 項（4-16）
Stall Messages	データベース・ユーザーのストール処理の概要	3.2.1.3 項（3-9）

表 2-5 Performance Monitor の各画面（続き）

画面	説明	参照先
Active User Stall Messages	現在のノード上にあるデータベースに接続されているすべてのプロセスを示し、ストールしたプロセスが存在する場合は、ストールが発生した理由も示す。	3.2.1.5 項（3-16）
Process Accounting	VMScluster 環境のローカル・プロセスに関して、常に更新されている OpenVMS アカウンティング情報を示す。	4.1.1.6 項（4-12）
Checkpoint Information	プロセスのチェックポイント情報	4.1.1.12 項（4-18）
Active User Chart	時間の経過ごとに、データベースに接続されているアクティブ・ユーザーの数を示す。	Performance Monitor のヘルプ
CPU Utilization	各データベース・プロセスの現在の CPU 使用率	3.3 項（3-28）
DBR Activity	ノート上でアクティブな状態の各データベース・リカバリ（DataBase Recovery：DBR）プロセスに関するオンライン情報を示す。	6.5.2 項（6-29）
Monitor Log	モニター・ログをオンラインで表示	Performance Monitor のヘルプ
Defined Logicals	Performance Monitor に現在アクセス可能な論理名をすべて一覧表示	2.3.1 項（2-72）
Lock Timeout History	タイムアウト・イベントの原因となっているオブジェクトを特定	3.8.1.5 項（3-58）
Lock Deadlock History	デッドロック・イベントの原因となっているオブジェクトを特定	3.8.1.4 項（3-57）
DBKEY Information	最後に取り出されたデータの dbkey、スナップショット、SPAM、AIP、ABM ページを示す。	3.2.1.4 項（3-15）
AIJ Statistics	After-image ジャーナルの概要	4.1.1.8 項（4-14）
Group Commit Statistics	AIJ グループ・コミット処理に関する情報	Performance Monitor のヘルプ

表 2-5 Performance Monitor の各画面（続き）

画面	説明	参照先
AIJ Journal Information	現在のノードにあるすべてのデータベースの After-image ジャーナルに関する情報	4.1.1.9 項（4-15）
AIJ Journal Growth Trend	所定の時間が経過する間、現在の AIJ のサイズをグラフで表示	Performance Monitor のヘルプ
ALS Statistics	AIJ ログ・サーバー・アクティビティ	Performance Monitor のヘルプ
2PC Statistics	分散トランザクション・パフォーマンスに関する情報	Performance Monitor のヘルプ
RUJ Statistics	アクティブな更新トランザクションすべてに関する情報を概要で示す。	Performance Monitor のヘルプ
Checkpoint Statistics	トランザクションとチェックポイント・アクティビティ	4.1.1.11 項（4-16）
Recovery Statistics	様々なリカバリ・フェーズを特定し、各フェーズが完了するまでにかかる時間を示す。	Performance Monitor のヘルプ
Hot Standby Statistics	ホット・スタンバイ機能のパフォーマンスとステータスに関する情報	Performance Monitor のヘルプ
Synchronization Mode Statistics	同期モードの各タイプの内訳	Performance Monitor のヘルプ
PIO Statistics--Data Fetches	データ・ページ・リクエストの処理に関する統計	4.1.1.3 項（4-9） 8.1.3 項（8-15）
PIO Statistics--SPAM Fetches	SPAM ページ・リクエストの処理に関する統計	4.1.1.4 項（4-10） 8.1.3 項（8-15）
PIO Statistics--Data Writes	データ・ファイルの書込みとバッファのマーク解除に関する情報を概要で示す。	4.1.1.2 項（4-8）
PIO Statistics--SPAM Writes	SPAM 書込み I/O 情報	Performance Monitor のヘルプ
PIO Statistics--File Access	ファイル・アクセスに関する情報	Performance Monitor のヘルプ
Asynchronous IO Statistics	データベース・ファイルへの非同期の読取りと書込みの統計を示す。	4.1.1.5 項（4-11）

表 2-5 Performance Monitor の各画面（続き）

画面	説明	参照先
IO Stall Time	すべての I/O ストール処理の概要を示す。	3.2.1.2 項（3-9）
GB Utilization	グローバル・バッファ内の各ページの使用率	Performance Monitor のヘルプ
GB Hot Page Information	グローバル・バッファ・プール内で、最も共有率の高いページの一覧を表示	Performance Monitor のヘルプ
GB Frequency Information	ページの共有について、グローバル・バッファ・プールの有効性を表示	Performance Monitor のヘルプ
IO Statistics (by file)	各データベース・ファイルの I/O 統計	4.2.1.4 項（4-137） 8.1.2.3 項（8-9）
Device Information	記憶領域デバイス情報をオンライン表示	Performance Monitor のヘルプ
Locking (one lock type)	指定した 1 つのロック・タイプに関する統計を示す。	3.8.1.3 項（3-56）
Locking (one stat field)	特定の統計フィールドについて、すべてのロック・タイプの統計を示す。	3.8.1.3 項（3-56）
Lock Statistics (by file)	記憶領域とスナップショット・ファイルで発生するページ・ロックに関する情報を示す。	Performance Monitor のヘルプ
Database Parameter Information	データベース・パラメータの変更を反映して、自動的に変更される動的情報を示す。	Performance Monitor のヘルプ
Row Cache (One Cache)	指定した行キャッシュに関する情報を概要で示す。	Performance Monitor のヘルプ
Row Cache (One Field)	特定の統計フィールドについて、すべての行キャッシュの行キャッシュ統計を示す。	Performance Monitor のヘルプ
Row Cache Utilization	指定した行キャッシュ内の各行について、使用率情報を示す。	Performance Monitor のヘルプ
Hot Row Information	指定した行キャッシュについて、アクセス頻度が最も高い行の一覧を表示	Performance Monitor のヘルプ

表 2-5 Performance Monitor の各画面（続き）

画面	説明	参照先
Row Cache Status	指定した行キャッシュに関して、全体的なステータスを示す。	Performance Monitor のヘルプ
Row Cache Queue Length	行キャッシュが CPU パフォーマンスに与える相対的な影響を示す情報を表示	Performance Monitor のヘルプ
Row Length Distribution	特定の行キャッシュに含まれる行の長さの分布を示す。	Performance Monitor のヘルプ
RCS Statistics	行キャッシュ・サーバー（Row Cache Server：RCS）プロセスのランタイム操作に関する情報を表示	Performance Monitor のヘルプ
Index Statistics (Retrieval)	ソート済みインデックスの取出しに関する情報	3.9.5.2 項（3-108） 8.1.3.3 項（8-28）
Index Statistics (Insertion)	ソート済みインデックスの変更に関する情報	3.9.5.2 項（3-108）
Index Statistics (Removal)	ソート済みインデックスの削除に関する情報	3.9.5.2 項（3-108）
Hash Index Statistics	ソート済みハッシュ・インデックスの更新と取出しに関する情報	3.9.5.2 項（3-108）
VM Usage Statistics	あるノード上のすべてのデータベース・ユーザーについて、仮想メモリーの動的な使用率を示す。	4.4.1 項（4-182）
Name translation	データベース・ダッシュボードの更新と論理名の変換に関する統計を示す。	Performance Monitor のヘルプ
Objects (one stat type)	特定のデータベース・オブジェクトの統計	3.4 項（3-30）
Objects (one stat field)	特定の収集カテゴリに関する統計	3.4 項（3-30）
Database Dashboard	データベースに接続しているプロセスが実際に使用しているデータベース・パラメータや属性の設定を示す。	2.2.15 項（2-58）
Online Analysis & Info.	分析情報を表示	2.2.16 項（2-60）

2.2.13 文字セル・インタフェースにおける書式付きバイナリ・ファイルへの統計出力

RMU Show Statistics コマンドで OUTPUT 修飾子を指定すると、デフォルトのファイル・タイプである .dat を使って、書式付きバイナリ・ファイルに統計を出力できます。

書式付きバイナリ出力ファイルを再生するには、INPUT 修飾子を指定して RMU Show Statistics コマンドを実行します。このファイルは、OUTPUT 修飾子を指定した RMU Show Statistics セッションであらかじめ作成しておく必要があります。再生モードについては、2.2.3 項（2-18）を参照してください。

INPUT 修飾子を使用すると、TIME 修飾子で再生表示の速度を変更できます。たとえば、TIME=60（1 分ごとに統計を収集）でセッションを記録し、TIME=1（1 秒ごとに表示を更新）で再生します。この再生速度では、10 時間の統計を 10 分で表示できます。1 秒あたりの表示速度は、元の時間（記録時間）を基準に計算されるので、再生速度は表示される数値に影響を与えることはありません。

OpenVMS VAX OpenVMS Alpha

OpenVMS では、次のコマンドを実行すると、database.stats という名前で書式付きバイナリ出力ファイルが作成されます。

```
$ RMU/SHOW STATISTICS/OUTPUT=database.stats mf_personnel.rdb
```

◆

Compaq Tru64 UNIX

Compaq Tru64 UNIX では、次のコマンドを実行すると、database.stats という名前の書式付きバイナリ出力ファイルが作成されます。

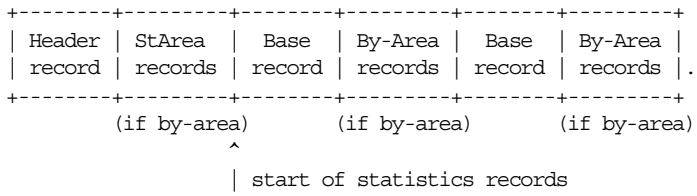
```
$ rmu -show statistics -output=database.stats mf_personnel.rdb
```

◆

バイナリ出力ファイルには、次の 4 つのレコード型が含まれます。

- ヘッダー・レコード（4096 バイト、固定長）。
- 記憶領域情報レコード（可変長レコードで均一にパックした 512 バイト構造）。
  各記憶領域情報レコードは、緊密にパックした複数（1 つ以上）の記憶領域エントリを含む。バイナリ出力ファイルのヘッダー・レコードは、アンパック・アルゴリズムの記述を含む。
- ベース統計レコード（4096 バイト、固定長）。
  各ベース統計レコードは、1 つのベース統計エントリを含む。アンパックは不要。
- 領域別情報レコード（可変長レコードで均一にパックした 512 バイト構造）。
  各領域別統計レコードは、緊密にパックした複数（1 つ以上）の記憶領域エントリを含む。バイナリ出力ファイルのヘッダー・レコードは、アンパック・アルゴリズムの記述を含む。

バイナリ出力ファイルは、次のようなフォーマットです。



レコード型には、型を識別できるタグはありません。各レコードは、ヘッダー・レコードに格納されているフラグや値に基づいた順序で格納されています。

OpenVMS VAX OpenVMS Alpha

OpenVMS では、次のコマンドを実行すると、書式付きバイナリ出力ファイルである database.stats の統計を Performance Monitor の Replay Mode を使用して表示できます。

```
$ RMU/SHOW STATISTICS/INPUT=database.stats
```

◆

OpenVMS VAX OpenVMS Alpha

Compaq Tru64 UNIX では、次のコマンドを実行すると、書式付きバイナリ出力ファイルである database.stats の統計を Performance Monitor の Replay Mode を使用して表示できます。

```
$ rmu -show statistics -input=database.stats
```

◆

OpenVMS VAX OpenVMS Alpha

ただし、異なるフォーマットでの出力が必要になる場合もあります。このような場合に対応するために、バイナリ・ファイルのフォーマットは次のテキスト・ファイルに収録されています。

```
SYS$LIBRARY:RMU$SHOW_STATISTICS.CDO
```

テキスト・ファイル RMU$SHOW_STATISTICS.CDO には、Performance Monitor バイナリ出力ファイルのすべてのレコードの一覧と定義が格納されています（複数の Oracle Rdb バージョンがインストールされている場合、テキスト・ファイルの名前は RMU$SHOW_STATISTICSvv.CDO となります。vv は、Oracle Rdb のバージョンです）。次の例は、SYS$LIBRARY:RMU$SHOW_STATISTICS.CDO ファイルを使って、バイナリ出力ファイルをデータ・リポジトリで定義する方法を示しています。オラクル社は、このファイルに含まれる定義を、専用のレポジトリに格納することをお薦めします。この例では、CDD$TOP.RDB$EXAMPLES という名前のリポジトリを使用します。

まず、新しいリポジトリを作成します。

```
$ REPOSITORY OPERATOR
CDO> DEFINE DIRECTORY CDD$TOP.RDB$EXAMPLES.
```

次に、フィールドとレコードを定義します。

```
CDO> SET DEFAULT CDD$TOP.RDB$EXAMPLES
CDO> @SYS$LIBRARY:RMU$SHOW_STATISTICS.CDO
```

Oracle Rdb の新バージョンでは、新しい統計が作成されています。Oracle Rdb では、新しいバージョンごとに SYS$LIBRARY:RMU$SHOW_STATISTICS.CDO ファイルが更新されます。Oracle RMU バイナリ統計を使用する既存のプロシージャは、Oracle Rdb の新バージョンでも動作します。ただし、Oracle Rdb の新しいバージョンで提供されている統計にアクセスするには、バイナリ・ファイルの定義とプログラムを、Oracle Rdb の新バージョンで更新する必要があります。◆

2.2.14 文字セル・インタフェースでの Performance Monitor のカスタマイズ

Performance Monitor では、カスタマイズした統計画面を作成できます。ベース統計情報（領域ごとでない統計やプロセスごとでない統計など）は、いずれもカスタマイズができます。

「Custom Statistics」画面のサイズによって、追加できる統計フィールドの数が決まります。「Custom Statistics」画面では、ヘッダーと水平メニューに 8 行使用するので、残りの行に最大 36 までの統計フィールドを追加できます。各統計フィールドは、画面上の別の行に表示されます。

「Custom Statistics」画面は通常の画面なので、情報は、ヒストグラムでのグラフィック表示や数値を含む表で表示できます。また、選択した統計フィールドでは、Time_plot and X_plot オプションを使用できます。

「Custom Statistics」画面を選択するには、「Menu」オプションを選択し、[↓] キーでカーソルを「Custom Statistics」オプションに移動してから [Enter] キーを押します。

初期設定では、「Custom Statistics」画面には、データベース・バインドとトランザクションという 2 つの統計が含まれています。この統計は、移動、削除、置換ができます。

「Custom Statistics」画面にフィールドを追加するには、既存の統計画面からヤンクして「Custom Statistics」画面へ暗黙的にプットする方法と、統計フィールドを手動で作成する方法の 2 つがあります。オラクル社は、簡単なヤンク・アンド・プットの方法をお薦めします。

統計フィールドをヤンク・アンド・プットするには、選択したい統計フィールドが含まれている統計画面の「Yank」メニュー・オプションを使用します。「Custom Statistics」画面には、「Yank」メニュー・オプションはありません。

「Y」を選択すると、選択可能なフィールドがメニューで表示されます。このメニューは、フィールドの「Help」メニューに類似しています。ヤンクしたい統計フィールドの文字を選択すると、そのフィールドは「Custom Statistics」画面へ自動的に追加されます。既に選択した統計フィールドは再度選択できないので、まず「Custom Statistics」画面から削除する必要があります。つまり、フィールドの選択メニューは、フィールドを選択するたびに更新されます。

選択したフィールドは、「Custom Statistics」画面で使用可能な最初の行に追加されます。ヤンク・アンド・プットの方法では、「Custom Statistics」画面上でフィールドを配置する行を

指定できません。ただし、フィールドを「Custom Statistics」画面にプットした後で、フィールドの位置を変更できます。

画面上で使用可能な統計フィールドをすべて選択した場合や、「Custom Statistics」画面で使用可能なフィールドがない場合、「Yank」メニュー・オプションは自動的にキャンセルされます。

「Custom Statistics」画面の作成には、ヤンク・アンド・プットの方法をお薦めします。既存の画面をブラウズし、監視したい統計フィールドを選択してください。

上級ユーザー向けには、統計フィールドを手動で作成する機能も提供されています。「Custom Statistics」画面の「Conifg」メニュー・オプションを選択すると、指定した統計フィールドのインデックスを、1 ～ 1020 の番号で明示的に入力できます。インデックスの詳細は、テキスト・ファイル SYS$LIBRARY:RMU$SHOW_STATISTICS.CDO を参照してください（複数の Oracle Rdb バージョンがインストールされている場合、テキスト・ファイルの名前は SYS$LIBRARY:RMU$SHOW_STATISTICSvv.CDO となります。vv は、Oracle Rdb のバージョンです）。統計ファイルの手動による選択では、既存の統計フィールドのインデックスを選択した場合、キーボードを押すとビープ音が鳴ります。ただし、エラー・メッセージは表示されません。

「Custom Statistics」画面で [C] を押すと、「Configuration」メニューが表示されます。「Configuration」メニューでは、「Custom Statistics」画面の統計フィールドの追加、削除、移動、圧縮ができます。「Configuration」メニューは、「Custom Statistics」画面の状態によって異なりますが、通常は 4 つのオプションがすべて表示されます。

「Add」メニュー・オプションを選択すると、プロンプトが表示されるので、統計フィールドを画面上に配置したい位置、統計フィールドのインデックス、統計フィールドのタイトルを選択します。指定したタイトルは、大カッコで囲んで各インデックスに追加されるので、どの統計を手動で選択したのかがわかります。

このオプションは、通常は表示されない統計を追跡したい場合に非常に便利です。たとえば、Performance Monitor の画面では、割当て仮想メモリー（Virtual Memory：VM）の総バイト数を追跡する統計フィールドは表示されません。インデックス番号 16 を選択すると、「Custom Statistics」画面に、割当て VM の総バイト数が表示されます。

「Delete」メニュー・オプションを選択すると、プロンプトが表示されるので、削除したい統計フィールドを選択します。

「Move」メニュー・オプションを選択すると、プロンプトが表示されるので、移動したい統計フィールドと、フィールドを表示する画面上の位置を選択します。

「Compress」メニュー・オプションを選択すると、画面の空白行をすべて削除します。このオプションは、1 つ以上の統計フィールドを削除した後に実行すると便利です。

初期設定では、「Custom Statistics」画面は次のように表示されます。

```
Node: TRIXIE         Oracle Rdb V7.0-00 Performance Monitor  3-MAY-1996 09:25:01
Rate: 3.00 Seconds             Custom Statistics            Elapsed: 00:00:21.23
Page: 1 of 1       RDBVMS_USER1:[LOGAN.V7]MF_PERSONNEL.RDB;1        Mode: Online
-------------------------------------------------------------------------------
statistic.........      rate.per.second.............. total....... average......
name..............      max..... cur..... avg....... count....... per.trans....
database binds                 0        0        0.0           1           0.0
transactions                   0        0        0.0           0           0.0
-------------------------------------------------------------------------------
Config Exit Graph Help Menu Options Pause Reset Set_rate Time_plot Unreset Write
```

初期設定では、データベース・バインドとトランザクション統計が表示されます。

1 つ以上のフィールドを「VM Usage Statistics」画面から「Custom Statistics」画面へコピーするには、「VM Usage Statistics」画面に進みます。

```
Node: TRIXIE         Oracle Rdb V7.0-00 Performance Monitor  3-MAY-1996 09:49:15
Rate: 3.00 Seconds             VM Usage Statistics          Elapsed: 00:24:35.29
Page: 1 of 1       RDBVMS_USER1:[LOGAN.V7]MF_PERSONNEL.RDB;1        Mode: Online
-------------------------------------------------------------------------------
statistic.........      rate.per.second.............. total....... average......
name..............      max..... cur..... avg....... count....... per.trans....
GET_VM calls                 187        0        0.2         351           0.0
FREE_VM calls                  0        0        0.0          14           0.0
GET_VM kilobytes             795        0        1.6        2402           0.0
FREE_VM kilobytes             32        0        0.7         978           0.0
$EXPREG calls                  0        0        0.0           0           0.0
-------------------------------------------------------------------------------
Exit Graph Help Menu Options Pause Reset Set_rate Time_plot Unreset Write X_plot
```

[Y] を押すと、「Yank-and-Put」メニューが表示されます。

```
Node: TRIXIE         Oracle Rdb V7.0-00 Performance Monitor  3-MAY-1996 09:51:56
Rate: 3.00 Seconds             VM Usage Statistics          Elapsed: 00:00:04.65
Page: 1 of 1       RDBVMS_USER1:[LOGAN.V7]MF_PERSONNEL.RDB;1        Mode: Online
-------------------------------------------------------------------------------
statistic.........      rate.per.second.............. total....... average......
name..............      max..... cur..... avg....... count....... per.trans....
A. GET_VM calls              74        0       72.9         339           0.0
B. FREE_VM calls              0        0        0.4           2           0.0
C. GET_VM kilobytes         343        0      336.3        1564           0.0
D. FREE_VM kilobytes         31        0       30.1         140           0.0
E. $EXPREG calls              0        0        0.0           0           0.0
-------------------------------------------------------------------------------
Type <return> or <letter> to select stats field, <control-Z> to cancel
```

[A] を押すと、「GET_VM calls」統計フィールドが選択されます。モードは選択モードのままですが、選択するオプションによって選択メニューは異なります。オプション A を選択すると、次のように新しいメニューが表示されます。

```
Node: TRIXIE         Oracle Rdb V7.0-00 Performance Monitor  3-MAY-1996 09:51:56
Rate: 3.00 Seconds            VM Usage Statistics           Elapsed: 00:00:04.65
Page: 1 of 1       RDBVMS_USER1:[LOGAN.V7]MF_PERSONNEL.RDB;1        Mode: Online
--------------------------------------------------------------------------------
statistic.........      rate.per.second.............. total....... average......
name..............      max..... cur..... avg....... count....... per.trans....
GET_VM calls                 74        0       72.9          339           0.0
A. FREE_VM calls              0        0        0.4            2           0.0
B. GET_VM kilobytes         343        0      336.3         1564           0.0
C. FREE_VM kilobytes         31        0       30.1          140           0.0
D. $EXPREG calls              0        0        0.0            0           0.0
--------------------------------------------------------------------------------
Type <return> or <letter> to select stats field, <control-Z> to cancel
```

「GET_VM calls」統計フィールドは、既に選択しているので再度選択できません。したがって、選択メニューからは除外されています。[B] を押すと、「GET_VM kilobytes」フィールドが選択されます。[Ctrl] キーを押しながら [Z] を押すと、「VM Usage Statistics」メニューは終了します。

次に、「Custom Statistics」画面に戻ってください。ヤンク・アンド・プットの方法で選択した「GET_VM calls」と「GET_VM kilobytes」フィールドが表示されています。

```
Node: TRIXIE         Oracle Rdb V7.0-00 Performance Monitor  3-MAY-1996 09:57:06
Rate: 3.00 Seconds            Custom Statistics             Elapsed: 00:05:13.90
Page: 1 of 1       RDBVMS_USER1:[LOGAN.V7]MF_PERSONNEL.RDB;1        Mode: Online
--------------------------------------------------------------------------------
statistic.........      rate.per.second.............. total....... average......
name..............      max..... cur..... avg....... count....... per.trans....
database binds                0        0        0.0            1           0.0
transactions                  0        0        0.0            0           0.0
GET_VM calls                 74        0        1.0          341           0.0
GET_VM kilobytes            343        0        5.4         1704           0.0
--------------------------------------------------------------------------------
Config Exit Graph Help Menu Options Pause Reset Set_rate Time_plot Unreset Write
```

表示されている統計フィールドを表示したくない場合は、[C] を押して「Config」メニュー・オプションを選択します。次のような「Configuration」メニューが表示されます。

```
Node: TRIXIE         Oracle Rdb V7.0-00 Performance Monitor  3-MAY-1996 10:01:22
Rate: 3.00 Seconds             Custom Statistics            Elapsed: 00:09:29.95
Page: 1 of 1       RDBVMS_USER1:[LOGAN.V7]MF_PERSONNEL.RDB;1        Mode: Online
-------------------------------------------------------------------------------
statistic.........      rate.per.second.............. total....... average......
name..............      max..... cur..... avg....... count....... per.trans....
database binds                 0        0        0.0           1           0.0
transactions                                                   0           0.0
GET_VM calls          +---Select Custom Configuration---+      343           0.0
GET_VM kilobytes      |                                 |     1843           0.0
                      |   A. Add a statistics field     |
                      |   B. Delete a statistics field  |
                      |   C. Move a statistics field    |
                      |   D. Compress the display       |
                      |                                 |
                      +---------------------------------+
-------------------------------------------------------------------------------
Type <return> or <letter> to select customization choice, <control-Z> to cancel
```

既存の統計フィールドを削除するには、メニュー・オプション [B] を選択します。既存の統計フィールドの一覧が表示されるので、削除したいフィールドを選択してください。

操作が完了すると、「Custom Statistics」画面が再度表示されます。

```
Node: TRIXIE         Oracle Rdb V7.0-00 Performance Monitor  3-MAY-1996 10:21:51
Rate: 3.00 Seconds             Custom Statistics            Elapsed: 00:29:59.74
Page: 1 of 1       RDBVMS_USER1:[LOGAN.V7]MF_PERSONNEL.RDB;1        Mode: Online
-------------------------------------------------------------------------------
statistic.........      rate.per.second.............. total....... average......
name..............      max..... cur..... avg....... count....... per.trans....
database binds                 0        0        0.0           1           0.0
transactions                   0        0        0.0           0           0.0
GET_VM kilobytes             343        0        1.5        2682           0.0
-------------------------------------------------------------------------------
Config Exit Graph Help Menu Options Pause Reset Set_rate Time_plot Unreset Write
```

削除した統計フィールドは、空白になっています。空白行は、自動的に削除されません。メニューで [C] を押して「Config」メニューを選択し、「Custom Configuration」メニューの [D] を選択すると、空白行を圧縮（削除）できます。

2.2.15 Performance Monitor の Database Dashboard 機能

Performance Monitor では、グローバル・データベース・プロセスと個別のデータベース・プロセスの両方に、Database Dashboard 機能が提供されています。この機能を実行するには、Performance Monitor のメイン・メニューから「Database Dashboard」オプションを選択してください。次のようなサブメニューが表示されます。

```
+----------- Select Display -----------+
|                                      |
|  A. IO Dashboard                     |
|  B. Locking Dashboard                |
|  C. AIJ Dashboard                    |
|  D. Checkpoint Dashboard             |
|  E. Hot Standby Dashboard            |
|  F. Row Cache Dashboard              |
|  G. RUJ Dashboard                    |
|  H. Monitor Dashboard                |
|  I. ABS Dashboard                    |
|  J. ALS Dashboard                    |
|  K. DBR Dashboard                    |
|  L. RCS Dashboard                    |
|  M. Per-Process I/O Dashboard        |
|  N. Per-Process Journal Dashboard    |
|  O. Per-Process Row Cache Dashboard  |
|                                      |
+--------------------------------------+
```

「Row Cache Dashboards」は、データベースのキャッシングが有効化されている場合にのみ表示されます。

Database Dashboard 機能は、データベースに接続しているプロセスが使用している実際のデータベース・パラメータと属性設定を表示します。この機能によって、データベース管理者（DataBase Administrator：DBA）は、実行時の論理名や構成パラメータの値、その他データベース属性の設定内容を確認できます。

OpenVMS VAX OpenVMS Alpha

オプションで、1 つのノード上でデータベースのパラメータや属性を、実行時に動的に更新できます。構成を変更した場合の影響を、データベース・プロセスを再起動することなく、実行時に確認できます。この更新は、永続的ではありません。

Database Dashboard 機能では、データベースの処理速度を上下させ、データベースの設定変更による影響をすぐに確認できます。

[U] を押して「Update」メニュー・オプションを選択すると、データベース属性の値を変更できます。データベース属性を更新するには、OPTIONS=UPDATE 修飾子を指定して Performance Monitor セッションを起動しておく必要があり、OpenVMS WORLD、BYPASS および SYSNAM 権限が必要です。

注意：「Update」オプションは、慎重に使用してください。Oracle Rdb は、更新された値に対してエラー・チェックを行いません。

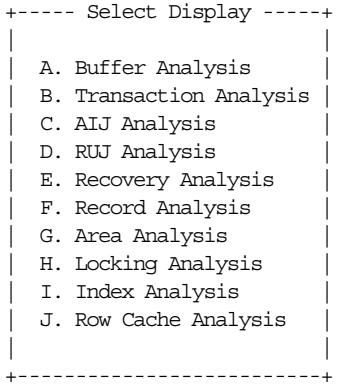
属性の更新内容は、データベース・ルート・ファイルには格納されないので注意してください。属性の更新は、データベースの変更による影響をテストおよび測定することを目的にしているので、データベース属性の永続的な変更は、対話型 SQL を使ってください。

データベース属性が更新されるのは、現行ノードのみです。◆

「Database Dashboard」画面とフィールドの詳細は、Performance Monitor のヘルプを参照してください。

2.2.16 Performance Monitor の Online Analysis 機能

Performance Monitor の Online Analysis 機能は、DBA がパフォーマンス上の問題を調査する際に重要な項目を特定します。この機能を実行するには、Performance Monitor のメイン・メニューの「Online Analysis & Info.」オプションを選択します。次のようなサブメニューが表示されます。



「Online Analysis」画面は、パフォーマンス上の問題を必ずしも示しているわけではありません。表示されている項目は、パフォーマンスの問題を示している可能性がありますが、ほとんどの場合は、更に調査する必要があります。

「Online Analysis」画面では、対話型 SQL で指定した実際のデータベース・パラメータと属性を使用します。Database Dashboard の属性は一時的な設定であるため、この画面では Database Dashboard 機能は使用しません。したがって、Database Dashboard を使ってオンラインで変更した内容は、分析結果には反映されません。ただし、データベース属性の設定のテスト方法としては、Database Dashboard 機能をお薦めします。永続的な属性設定を行うには、対話型 SQL を使用してください。

Online Analysis 機能は、指定した画面のリフレッシュ間隔で起動します。Online Analysis 機能は、かなり CPU 集中型であり、非常に短い時間で実行しても分析結果が大きく変動することはほとんどないため、オラクル社は、リフレッシュ間隔に 3 秒以上を設定することをお薦めします。

「Online Analysis」画面は、OUTPUT 修飾子を使って作成したバイナリ出力ファイルには記録されません。したがって、INPUT 修飾子を使ってバイナリ・ファイルを再生しても、この画面は参照できません。

「Online Analysis」画面の詳細は、Performance Monitor のヘルプを参照してください。

2.3 Oracle Rdb の論理名と構成パラメータ

この項では、Oracle Rdb の論理名と構成パラメータを使ってデータベース・パフォーマンスを向上する方法について説明します。表 2-6（2-61）は、論理名と構成パラメータの一覧と、データベースのチューニングで使用する方法の簡単な説明です。付録 A（A-1）には、論理名と構成パラメータの詳細と使用方法が掲載されています。

表 2-6 Oracle Rdb の論理名と構成パラメータの概要

論理名 構成パラメータ	機能
RDB$CHARACTER_SET Compaq Tru64 UNIX は適用外	Oracle Rdb で使用する代替キャラクタ・セットを定義する。
RDB$LIBRARY RDB_LIBRARY	外部関数などの外部ルーチン・イメージの格納に使用可能な保護付きライブラリを指定する。
RDB$RDBSHR_EVENT_FLAGS Compaq Tru64 UNIX は適用外	LIB$GET_EF システム・サービスが、起動時に RDB$SHARE に割り当てた 4 つのイベント・フラグ番号を上書きする。
RDB$REMOTE_BUFFER_SIZE SQL_NETWORK_BUFFER_SIZE	ネットワーク転送のデフォルト・バッファ・サイズを、システム・クォータの上限の範囲内で変更する。
RDB$REMOTE_MULTIPLEX_OFF SQL_NETWORK_NUMBER_ATTACHES	同じノードに対する複数のリモート・データベース・アクセスに使用するリモート・サーバー・プロセスの数を制御する。
RDB$ROUTINES RDB_ROUTINES	外部ルーチン・イメージのロケーションを指定する。
RDBVMS$CREATE_DB RDB_CREATE_DB	SQL CREATE DATABASE 文を使用したデータベースの作成を許可するユーザーを指定する。

表 2-6 Oracle Rdb の論理名と構成パラメータの概要（続き）

論理名 構成パラメータ	機能
RDM$BIND_ABS_LOG_FILE RDB_BIND_ABS_LOG_FILE	After-image ジャーナル・バックアップ・サーバー（After-image journal Backup Server：ABS）ログ・ファイルのファイル名を指定する。
RDM$BIND_ABS_OVERWRITE_ALLOWED RDB_BIND_ABS_OVERWRITE_ALLOWED	After-image ジャーナル・バックアップ・サーバーが上書きされた AIJ をリセットするかどうかを指定する。
RDM$BIND_ABS_OVERWRITE_IMMEDIATE RDB_BIND_ABS_OVERWRITE_IMMEDIATE	ジャーナルを即時にリセットするかどうかを指定する。RDM$BIND_ABS_OVERWRITE_ALLOWED または RDB_BIND_ABS_OVERWRITE_ALLOWED の有効化が必要。
RDM$BIND_ABS_QUIET_POINT RDB_BIND_ABS_QUIET_POINT	After-image ジャーナル・バックアップ・サーバーが静止ポイント・ジャーナル・バックアップを実行するかどうかを示す。
RDM$BIND_ABW_ENABLED RDB_BIND_ABW_ENABLED	非同期バッチ書込み操作の有効化または無効化。
RDM$BIND_AIJ_CHECK_CONTROL_RECS RDB_BIND_AIJ_CHECK_CONTROL_RECS	Oracle Rdb が、AIJ キャッシュ・フォーマット中に制御レコードをチェックするかどうかを指定する。
RDM$BIND_AIJ_EMERGENCY_DIR RDB_BIND_AIJ_EMERGENCY_DIR	緊急 AIJ のロケーションを指定する。
RDM$BIND_AIJ_IO_MAX RDB_BIND_AIJ_IO_MAX	AIJ グループ・コミットの最大 I/O バッファ・サイズのデフォルト値の上書きを許可する。デフォルト値は、127 ブロック。
RDM$BIND_AIJ_IO_MIN RDB_BIND_AIJ_IO_MIN	AIJ グループ・コミットの最小 I/O バッファ・サイズの上書きを許可する。デフォルト値は、8 ブロック。
RDM$BIND_AIJ_STALL RDB_BIND_AIJ_STALL	.aij ログ・ファイルへコミット・レコードをサブミットした後、トランザクションが待機する時間をミリ秒で定義する。

表 2-6 Oracle Rdb の論理名と構成パラメータの概要（続き）

論理名 構成パラメータ	機能
RDM$BIND_AIJ_SWITCH_GLOBAL_CKPT RDB_BIND_AIJ_SWITCH_GLOBAL_CKPT	AIJ 切替えが発生した後に、Oracle Rdb がグローバル・チェックポイントを実行するかどうかを指定する。
RDM$BIND_ALS_CREATE_AIJ RDB_BIND_ALS_CREATE_AIJ	AIJ 切替えが一時停止状態になった場合に、ALS サーバーが緊急 AIJ を作成するかどうかを指定する。
RDM$BIND_APF_DEPTH RDB_BIND_APF_DEPTH	Oracle Rdb がプロセスの非同期プリフェッチを実行する際のバッファの数（深さ）を指定する。
RDM$BIND_APF_ENABLED RDB_BIND_APF_ENABLED	非同期プリフェッチの有効化または無効化。
RDM$BIND_BATCH_MAX RDB_BIND_BATCH_MAX	バッチ書込みまたは非同期バッチ書込みの一部として、データベースに書き込まれるライブ・データ・キャッシュ・バッファの数を定義する。この論理名または構成パラメータを、ユーザーに割り当てられるキャッシュ・バッファの数よりも小さい値に設定することにより、バッチ書込みで発生する I/O バッファのサイズを制御できる。これにより、システムの I/O 処理の予測と均一化が可能。
RDM$BIND_BUFFERS RDB_BIND_BUFFERS	実行時のバッファの代替数を指定できる。特定のタスクで使用するバッファの数のデフォルト値を一時的に上書きし、その他ではデフォルト値を使用したい場合に便利。
RDM$BIND_BUFOBJ_ENABLED Compaq Tru64 UNIX は適用外	OpenVMS Alpha バッファ・オブジェクトを有効にし、Oracle Rdb ローカル・バッファを物理メモリー内にロックするかどうかを指定する。
RDM$BIND_CBL_ENABLED RDB_BIND_CBL_ENABLED	粗いバッファ・ロックを有効にするかどうかを指定する。
RDM$BIND_CKPT_BLOCKS RDB_BIND_CKPT_BLOCKS	チェックポイントが発生するまでの AIJ ブロックの数を指定する。
RDM$BIND_CKPT_TIME RDB_BIND_CKPT_TIME	チェックポイントが発生するまでの時間を秒単位で指定する。

表 2-6 Oracle Rdb の論理名と構成パラメータの概要（続き）

論理名 構成パラメータ	機能
RDM$BIND_CKPT_TRANS_INTERVAL RDB_BIND_CKPT_TRANS_INTERVAL	コミットしたトランザクションの数を基準に、プロセスごとのチェックポイント間隔の値を定義する。高速コミット処理の有効化が必要。
RDM$BIND_CLEAN_BUF_CNT RDB_BIND_CLEAN_BUF_CNT	置換するために、プロセスの最も古いバッファのキューの最後に配置しておくクリーン・バッファの数を指定する。この論理名または構成パラメータに適切な値を指定することにより、プロセスで発生する I/O ストールの数を減らすことが可能。
RDM$BIND_COMMIT_STALL RDB_BIND_COMMIT_STALL	トランザクションが、グループ・コミット・プロセスとなるために、試行後に待機する時間をミリ秒で指定する。
RDM$BIND_DAPF_DEPTH_BUF_CNT RDB_BIND_DAPF_DEPTH_BUF_CNT	物理領域からプリフェッチするバッファの数を指定する。デフォルト値は、データベース・ユーザー向けに定義したバッファの数の 1/2。
RDM$BIND_DAPF_ENABLED RDB_BIND_DAPF_ENABLED	検出した非同期プリフェッチの有効化または無効化。
RDM$BIND_DAPF_START_BUF_CNT RDB_BIND_DAPF_START_BUF_CNT	検出非同期プリフェッチ（Detected Asynchronous Prefetch：DAPF）読込み操作が開始する前に、物理領域から順次アクセスするバッファの数を指定する。
RDM$BIND_HRL_ENABLED RDM_BIND_HRL_ENABLED	ホールド読取りロックを有効にするかどうかを指定する。
RDM$BIND_LOCK_TIMEOUT_INTERVAL RDB_BIND_LOCK_TIMEOUT_INTERVAL	ロックが解除されるまでトランザクションが待機する時間を設定する。
RDM$BIND_MAX_DBR_COUNT RDB_BIND_MAX_DBR_COUNT	データベース・モニターが同時に起動するデータベース・リカバリ（DataBase Recovery：DBR）処理の最大数を定義する。
RDM$BIND_OPTIMIZE_AIJ_RECLEN RDB_BIND_OPTIMIZE_AIJ_RECLEN	AIJ レコード長の平均値を指定することにより、RMU Optimize コマンドのパフォーマンスを調整する。

表 2-6 Oracle Rdb の論理名と構成パラメータの概要（続き）

論理名 構成パラメータ	機能
RDM$BIND_RCACHE_INSERT_ENABLED RDB_BIND_RCACHE_INSERT_ENABLED	行キャッシュに行を挿入可能かどうかを示す。
RDM$BIND_RCACHE_RCRL_COUNT RDB_BIND_RCACHE_RCRL_COUNT	予約済のキャッシュ・スロットの数を示す。
RDM$BIND_RCS_BATCH_COUNT RDB_BIND_RCS_BATCH_COUNT	行キャッシュ・サーバー（Row Cache Server：RCS）が 1 つのバッチで削除する行の数を定義する。
RDM$BIND_RCS_CHECKPOINT RDB_BIND_RCS_CHECKPOINT	行キャッシュ・サーバーがチェックポイントを実行するかどうかを示す。
RDM$BIND_RCS_CKPT_BUFFER_CNT RDB_BIND_RCS_CKPT_BUFFER_CNT	チェックポイント操作の間、行キャッシュ・サーバープロセスが 1 つのバッチとして検査するバッファの数を示す。
RDM$BIND_RCS_LOG_FILE RDB_BIND_RCS_LOG_FILE	行キャッシュ・サーバー（RCS）ログ・ファイルのファイル名を定義する。
RDM$BIND_RCS_MAX_COLD RDB_BIND_RCS_MAX_COLD	マーク・レコードの数を示し、これを上回ると行キャッシュ・サーバーは削除を開始する。
RDM$BIND_RCS_MIN_COLD RDB_BIND_RCS_MIN_COLD	マーク解除レコードの数を示し、これを下回ると行キャッシュ・サーバーは削除を完了する。
RDM$BIND_RCS_SWEEP_INTERVAL RDB_BIND_RCS_SWEEP_INTERVAL	RCS を削除する間隔を秒で示す。
RDM$BIND_READY_AREA_SERIALLY RDB_BIND_READY_AREA_SERIALLY	Oracle Rdb は、ロックが要求された順序で論理および物理領域にロック要求を付与する。これにより、ロック・スタベーションの発生を防止できる。
RDM$BIND_RUJ_ALLOC_BLKCNT RDB_BIND_RUJ_ALLOC_BLKCNT	.ruj ファイルのデフォルト値を上書きする。デフォルトでは、.ruj ファイルは 127 ブロックで作成される。
RDM$BIND_RUJ_EXTEND_BLKCNT RDB_BIND_RUJ_EXTEND_BLKCNT	データベースを使用する各プロセス向けに、.ruj ファイルを事前に拡張する。ブロック・カウントは、0 ～ 9999（未満）の値を定義でき、デフォルトは 100。

表 2-6 Oracle Rdb の論理名と構成パラメータの概要（続き）

論理名 構成パラメータ	機能
RDM$BIND_SNAP_QUIET_POINT RDB_BIND_SNAP_QUIET_POINT	スナップショット・トランザクションが静止ロックを保留し、データベースまたは AIJ バックアップをストールするかどうかを制御する。
RDM$BIND_STATS_AIJ_ARBS_PER_IO RDB_BIND_STATS_AIJ_ARBS_PER_IO	AIJ I/O あたりの AIJ 要求ブロックのデフォルト値を上書きする。デフォルト値は、2。
RDM$BIND_STATS_AIJ_BKGRD_ARB_RATIO RDB_BIND_STATS_AIJ_BKGRD_ARB_RATIO	バックグラウンド AIJ 要求ブロックのしきい値のデフォルトを上書きする。デフォルト値は、50。
RDM$BIND_STATS_AIJ_BLKS_PER_IO RDB_BIND_STATS_AIJ_BLKS_PER_IO	AIJ I/O あたりのブロックのデフォルト値を上書きする。デフォルト値は、2。
RDM$BIND_STATS_AIJ_SEC_TO_EXTEND RDB_BIND_STATS_AIJ_SEC_TO_EXTEND	AIJ を拡張する秒数のデフォルト値を上書きする。デフォルト値は、60。
RDM$BIND_STATS_BTR_FETCH_DUP_RATIO RDB_BIND_STATS_BTR_FETCH_DUP_RATIO	B ツリーの重複フェッチのしきい値のデフォルトを上書きする。デフォルトのしきい値は、15。
RDM$BIND_STATS_BTR_LEF_FETCH_RATIO RDB_BIND_STATS_BTR_LEF_FETCH_RATIO	B ツリーのリーフ・ノード・フェッチのしきい値のデフォルトを上書きする。デフォルトのしきい値は、25。
RDM$BIND_STATS_DBR_RATIO RDB_BIND_STATS_DBR_RATIO	DBR を起動するしきい値のデフォルトを上書きする。デフォルトのしきい値は、15。
RDM$BIND_STATS_ENABLED RDB_BIND_STATS_ENABLED	プロセスについて、データベース統計の書込みを有効化または無効化。デフォルトでは、書込みは有効。
RDM$BIND_STATS_FULL_BACKUP_INTRVL RDB_BIND_STATS_FULL_BACKUP_INTRVL	フル・データベース・バックアップのしきい値を上書きする。デフォルトのしきい値は、6。
RDM$BIND_STATS_GB_IO_SAVED_RATIO RDB_BIND_STATS_GB_IO_SAVED_RATIO	GB IO 保存のデフォルトのしきい値を上書きする。デフォルトのしきい値は、85。
RDM$BIND_STATS_GB_POOL_HIT_RATIO RDB_BIND_STATS_GB_POOL_HIT_RATIO	GB プール・ヒットのデフォルトのしきい値を上書きする。デフォルトのしきい値は、85。

表 2-6 Oracle Rdb の論理名と構成パラメータの概要（続き）

論理名 構成パラメータ	機能
RDM$BIND_STATS_LB_PAGE_HIT_RATIO RDB_BIND_STATS_LB_PAGE_HIT_RATIO	LB/AS ページ・ヒットのデフォルトのしきい値を上書きする。デフォルト値は、75。
RDM$BIND_STATS_MAX_HASH_QUE_LEN RDB_BIND_STATS_MAX_HASH_QUE_LEN	ハッシュ・テーブルのキュー長のデフォルトのしきい値を上書きする。デフォルトのしきい値は、2 行。
RDM$BIND_STATS_MAX_LOCK_STALL RDB_BIND_STATS_MAX_LOCK_STALL	ロック・ストールのデフォルトのしきい値を上書きする。デフォルトのしきい値は、2。
RDM$BIND_STATS_MAX_TX_DURATION RDB_BIND_STATS_MAX_TX_DURATION	トランザクション継続時間のデフォルトのしきい値を上書きする。デフォルト値は、15。
RDM$BIND_STATS_PAGES_CHECKED_RATIO RDB_BIND_STATS_PAGES_CHECKED_RATIO	チェック済ページのデフォルトのしきい値を上書きする。デフォルトのしきい値は、10 ページ。
RDM$BIND_STATS_RECS_FETCHED_RATIO RDB_BIND_STATS_RECS_FETCHED_RATIO	プリフェッチ済レコードのデフォルトのしきい値を上書きする。デフォルトのしきい値は、20 レコード。
RDM$BIND_STATS_RECS_STORED_RATIO RDB_BIND_STATS_RECS_STORED_RATIO	格納済レコードのデフォルトのしきい値を上書きする。デフォルトのしきい値は、20 レコード。
RDM$BIND_STATS_RUJ_SYNC_IO_RATIO RDB_BIND_STATS_RUJ_SYNC_IO_RATIO	同期 RUJ I/O のデフォルトのしきい値を上書きする。デフォルトのしきい値は、10。
RDM$BIND_STATS_VERB_SUCCESS_RATIO RDB_BIND_STATS_VERB_SUCCESS_RATIO	命令の成功率のデフォルトのしきい値を上書きする。デフォルトのしきい値は、25。
RDM$BIND_SYSTEM_BUFFERS_ENABLED Compaq Tru64 UNIX または OpenVMS VAX では適用外	システム領域のグローバル・セクションが使用されているかどうかを示す。
RDM$BIND_TSN_INTERVAL RDB_BIND_TSN_INTERVAL	ジャーナルへのコミットの最適化を有効にした場合、1 つのバッチとして割り当てるトランザクションの数を示す。
RDM$BIND_VM_SEGMENT RDB_BIND_VM_SEGMENT	メモリーの断片化を防ぐのに必要となるバイト数を割り当てる。

表 2-6 Oracle Rdb の論理名と構成パラメータの概要（続き）

論理名 構成パラメータ	機能
RDM$BUGCHECK_DIR RDB_BUGCHECK_DIR	bugcheck ファイルを、デフォルト・ディレクトリから別のロケーションにリダイレクトする。現行のデフォルト・ログイン・ディレクトリに、bugcheck ダンプ・ファイル用の十分な領域がない場合に便利。
RDM$BUGCHECK_IGNORE_FLAGS RDB_BUGCHECK_IGNORE_FLAGS	bugcheck ダンプ・ファイルのサイズを縮小する。
RDM$MAILBOX_CHANNEL Compaq Tru64 UNIX は適用外	データベース・モニター・メールボックスのノード固有のアドレスを示す。このアドレスは、プロセスが適切なデータベース・モニターと通信する際に使用する。
RDM$MONITOR RDB_MONITOR	Oracle Rdb モニター・ログ・ファイルを格納するデバイスとディレクトリを定義する。この論理値には、ファイル名の指定は含めない。論理名または構成パラメータで定義するディレクトリ・ロケーションは、RMONSTART.COM コマンド・ファイルのみがテストおよび使用する。
RDM$MON_USERNAME Compaq Tru64 UNIX は適用外	起動時に、モニター・プロセスがクォータを継承するユーザーの名前を指定する。この論理名により、グローバル・バッファを有効にした場合に、起動時に発生する可能性がある超過クォータ・エラーを防止できる。
RDMS$AUTO_READY RDB_AUTO_READY	プロセスが、CU モードで論理領域のキャリーオーバー・ロックを既に保持している場合、CR モードで論理領域ロックを要求しても、CU モードでのロック取得を許可する。
RDMS$BIND_OUTLINE_FLAGS RDB_BIND_OUTLINE_FLAGS	これにより、Oracle Rdb は、クエリー・アウトラインを無視する。
RDMS$BIND_OUTLINE_MODE RDB_BIND_OUTLINE_MODE	クエリーに複数のアウトラインが存在する場合、この論理名によって、使用するアウトラインを選択する。

表 2-6 Oracle Rdb の論理名と構成パラメータの概要（続き）

論理名 構成パラメータ	機能
RDMS$BIND_PRESTART_TXN RDB_BIND_PRESTART_TXN	アプリケーションの外部で事前に起動したトランザクションのデフォルト設定を行う。
RDMS$BIND_QG_CPU_TIMEOUT RDB_BIND_QG_CPU_TIMEOUT	クエリーの実行を最適化するために使用する CPU 時間を制限する。ユーザーがクエリーを入力して、この値で設定した CPU 経過時間を超えた場合、ユーザーにエラー・メッセージを送信し、クエリーを強制終了する。
RDMS$BIND_QG_REC_LIMIT RDB_BIND_QG_REC_LIMIT	クエリーが返す行数に対して、プロセスまたはシステムの上限を設定する。ユーザーがクエリーを入力して、この値で設定した行数を超えた場合、ユーザーにエラー・メッセージを送信し、クエリーを強制終了する。
RDMS$BIND_QG_TIMEOUT RDB_BIND_QG_TIMEOUT	オプティマイザがクエリーのコンパイルにかける時間に対して、システムの上限を設定する。ユーザーがクエリーを入力して、この値で設定した経過時間を超えた場合、ユーザーにエラー・メッセージを送信し、クエリーを強制終了する。
RDMS$BIND_SEGMENTED_STRING_BUFFER RDB_BIND_SEGMENTED_STRING_BUFFER	セグメント化した文字列を操作する場合の、I/O 操作のオーバーヘッドを軽減する。
RDMS$BIND_SEGMENTED_STRING_COUNT RDB_BIND_SEGMENTED_STRING_COUNT	セグメント化した文字列 ID リストでの、割当てサイズをエントリ数で指定する。ID リストは、テーブル行のセグメント化文字列のマテリアライズと操作に使用する。この論理名または構成パラメータを定義することにより、インポート・オペレーションの失敗やプロセス・ループを防止できる。
RDMS$BIND_SEGMENTED_STRING_DBKEY_SCOPE RDB_BIND_SEGMENTED_STRING_DBKEY_SCOPE	プロセスが、変更したセグメント化文字列の dbkey を再利用しているかどうかを示す。

表 2-6 Oracle Rdb の論理名と構成パラメータの概要（続き）

論理名 構成パラメータ	機能
RDMS$BIND_SORT_WORKFILES RDB_BIND_SORT_WORKFILES	作業ファイルが必要な場合に、使用する作業ファイル SORT の数を指定する。
RDMS$BIND_VALIDATE_CHANGE_FIELD RDB_BIND_VALIDATE_CHANGE_FIELD	SQL ALTER DOMAIN 文で、データ・レコードを検査し、これを新しいメタデータ定義に変換する。
RDMS$BIND_WORK_FILE RDB_BIND_WORK_FILE	RDMS$BIND_WORK_VM または RDB_BIND_WORK_VM と一緒に使用する場合、マッチング操作でのディスク I/O のオーバーヘッドを軽減する。
RDMS$BIND_WORK_VM RDB_BIND_WORK_VM	テンポラリ・テーブルを使用するマッチング操作について、マッチング操作で使用するためにプロセスに割り当てる仮想メモリー（Virtual Memory：VM）のサイズを指定することにより、ディスク I/O のオーバーヘッドを軽減する。
RDMS$DEBUG_FLAGS RDB_DEBUG_FLAGS	データベース・アクセス・ストラテジを検査し、プログラムの実行時にかかるコストを試算する。
RDMS$DEBUG_FLAGS_OUTPUT RDB_DEBUG_FLAGS_OUTPUT	プログラム実行時に、RDMS$DEBUG_FLAGS または RDB_DEBUG_FLAGS からの出力を収集する出力ファイルの名前を指定する。
RDMS$DIAG_FLAGS RDB_DIAG_FLAGS	エラーのあるクエリーの検出に使用可能。
RDMS$KEEP_PREP_FILES Compaq Tru64 UNIX は適用外	これにより、RDBPRE プリプロセッサが、中間マクロ（.mar）と言語ファイルを保持する。デフォルトでは、プリプロセッサがソース・ファイルの処理を完了した時点で、ファイルを削除する。
RDMS$RUJ RDB_RUJ	.rdb ファイルのロケーションとは別のディスクやディレクトリに .ruj ファイルを配置し、データベースによるディスクやディレクトリの競合を軽減する。
RDMS$USE_OLD_CONCURRENCY RDB_USE_OLD_CONCURRENCY	Oracle Rdb V4.1 で採用された分離レベルの使用をアプリケーションに許可する。

表 2-6 Oracle Rdb の論理名と構成パラメータの概要（続き）

論理名 構成パラメータ	機能
RDMS$USE_OLD_COST_MODEL RDB_USE_OLD_COST_MODEL	これにより、オプティマイザは、作業負荷統計または記憶領域統計を使用しない。
RDMS$USE_OLD_COUNT_RELATION RDB_USE_OLD_COUNT_RELATION	空のテーブルでの CREATE INDEX の最適化を無効化。
RDMS$USE_OLD_SEGMENTED_STRING RDB_USE_OLD_SEGMENTED_STRING	古いフォーマット（連鎖）セグメント化文字列をデフォルトで保持する。新旧フォーマットのセグメント化文字列の混在をサポート。
RDMS$USE_OLD_UPDATE_RULES RDB_USE_OLD_UPDATE_RULES	RDO を使用するアプリケーションに対して、古い更新規則を使用する。バージョン 4.0 以下の Oracle Rdb では、リレーションを他のリレーションと結合すると、そのリレーションに含まれるレコードの変更や削除は許可しない。この論理名により、アプリケーションを変更するまでの間、Oracle Rdb バージョン 4.0 以下での RDO 更新を維持できる。
RDMS$VALIDATE_ROUTINE RDB_VALIDATE_ROUTINE	無効なルーチンを有効としてマークする。プロセスが RDMS$VALIDATE_ROUTINE または RDB_VALIDATE_ROUTINE に 1 を割り当てる場合、プロセスが読取り / 書込みトランザクション内でプロシージャをコールするとき、Oracle Rdb は、無効なルーチンを有効としてマークする。
RDO$EDIT Compaq Tru64 UNIX は適用外	対話型 RDO クエリーの編集に使用するシステム・エディタを示す。
RDOINI Compaq Tru64 UNIX は適用外	RDO の初期化情報を格納しているファイルの名前を指定する。論理名が存在し、かつ指定したファイルが存在する場合、RDO は、このファイル内のコマンドを最初に実行してから、RDO プロンプトを表示し、入力コマンドを受け取る。

表 2-6 Oracle Rdb の論理名と構成パラメータの概要（続き）

論理名 構成パラメータ	機能
RMU$EDIT Compaq Tru64 UNIX は適用外	Performance Monitor ツール機能のノートパッドの編集に使用するシステム・エディタを示す。有効な値は、EDT、LSE、TPU。デフォルトのテキスト・エディタは、EDT。
SQL$DATABASE SQL_DATABASE	データベースを明示的に宣言しない場合に、SQL が宣言するデータベースを指定する。
SQL$DISABLE_CONTEXT Compaq Tru64 UNIX は適用外	2 フェーズ・コミット・プロトコルを無効化。これは、バッチ更新トランザクションを実行したい場合に、分散トランザクションを無効にするのに便利。
SQL$EDIT Compaq Tru64 UNIX は適用外	対話型 SQL クエリーの編集に使用するシステム・エディタを示す。
SQLINI Compaq Tru64 UNIX は適用外	SQL の初期化情報を格納しているファイルの名前を指定する。論理名が存在し、かつ指定したファイルが存在する場合、SQL は、このファイル内のコマンドを最初に実行してから、SQL プロンプトを表示し、入力コマンドを受け取る。
SQL$KEEP_PREP_FILES SQL_KEEP_PREP_FILES	これにより、SQL プリコンパイラまたは SQL モジュール言語が、中間マクロ（.mar）と言語ファイルを保持する。デフォルトでは、プリコンパイラがソース・ファイルの処理を完了した時点で、ファイルを削除する。

2.3.1 文字セル・インタフェースにおける Performance Monitor の「Defined Logicals」画面

OpenVMS VAX OpenVMS Alpha

「Defined Logicals」画面では、Performance Monitor に現在アクセス可能なすべての論理名、論理名を定義しているテーブルの名前、対応付けられた論理定義が表示されます。

「Defined Logicals」画面にアクセスするには、メイン・メニューから「Process Information」オプションを選択し、サブメニューの「Defined Logicals」を選択します。

論理名は、わかりやすいように、接尾辞のアルファベット順に表示されています。つまり、論理名は、機能名（RDMS$、RDM$、SQL$ など）を取り除いた状態でソートされています。機能名の接頭辞は不明な場合が多いため、この方法は非常に便利です。

「Defined Logicals」画面は、Brief モードと Full モードの 2 つのモードで表示できます。デフォルトは、Brief モードです。[B]（メニューに「Brief」と表示）を押すと、Brief モードを選択できます。Brief モードでは、実際に定義され、アクセス可能な論理名のみが「Defined Logicals」画面に表示されます。表示されているテーブル名は、論理名が定義されている実際のテーブルであり、定義は、論理名に定義されている値です。新しい論理名がシステム上で定義されるたび、画面は更新されます。次に、Brief モードで表示された「Defined Logicals」画面を示します。

```
Node: TRIXIE         Oracle Rdb V7.0-00 Performance Monitor 17-MAY-1996 14:46:12
Rate: 3.00 Seconds              Defined Logicals            Elapsed: 05:47:55.00
Page: 1 of 3          SQL_DISK1:[USER]MF_PERSONNEL.RDB;1             Mode: Online
--------------------------------------------------------------------------------
Logical.Name..... Table.Name..... Logical.Definition.....
SQL$DATABASE                  LNM$PROCESS_TABLE   SQL_PERSONNEL
RDM$MAILBOX_CHANNEL           LNM$SYSTEM_TABLE    MBA64:
RDM$MONITOR                   LNM$SYSTEM_TABLE    SYS$SYSROOT:[SYSEXE]
--------------------------------------------------------------------------------
Exit Full Help Menu >next_page <prev_page Set_rate Write !
```

[F]（メニューに「Full」と表示）を押すと、Full モードを選択できます。Full モードでは、論理名が定義済がどうかにかかわらず、すべての論理名が「Defined Logicals」画面に表示されます。論理名が定義されている場合は、Brief モードと同様に、テーブル名と論理名が表示されます。論理名が定義されていない場合は、テーブル名には、論理名の格納先となるテーブルの名前が表示され、定義は表示されません。Full モードは、論理名のスペルや、論理名が適切なテーブルで定義されているかどうかを確認するときに便利です。次は、Full モードで表示された「Defined Logicals」画面です。

```
Node: TRIXIE         Oracle Rdb V7.0-00 Performance Monitor 17-MAY-1996 14:48:03
Rate: 3.00 Seconds              Defined Logicals            Elapsed: 05:50:17.67
Page: 1 of 5          SQL_DISK1:[USER]MF_PERSONNEL.RDB;1            Mode: Online
--------------------------------------------------------------------------------
Logical.Name................. Table.Name......... Logical.Definition..........
RDMS$AUTO_READY              LNM$FILE_DEV
RDM$BIND_ABS_QUIET_POINT     LNM$FILE_DEV
RDM$BIND_ABW_ENABLED         LNM$FILE_DEV
RDM$BIND_AIJ_IO_MAX          LNM$FILE_DEV
RDM$BIND_AIJ_STALL           LNM$FILE_DEV
RDM$BIND_APF_DEPTH           LNM$FILE_DEV
RDM$BIND_APF_ENABLED         LNM$FILE_DEV
RDM$BIND_BATCH_MAX           LNM$FILE_DEV
RDM$BIND_BUFFERS             LNM$FILE_DEV
RDM$BIND_CKPT_TRANS_INTERVAL LNM$FILE_DEV
RDM$BIND_CLEAN_BUF_CNT       LNM$FILE_DEV
RDM$BIND_COMMIT_STALL        LNM$FILE_DEV
RDM$BIND_DAPF_DEPTH_BUF_CNT  LNM$FILE_DEV
RDM$BIND_DAPF_ENABLED        LNM$FILE_DEV
RDM$BIND_DAPF_START_BUF_CNT  LNM$FILE_DEV
RDMS$BIND_EXEC_STACK_SIZE    LNM$FILE_DEV
RDMS$BIND_EXT_FILE_NAME_ONLY LNM$FILE_DEV
--------------------------------------------------------------------------------
Brief Exit Help Menu >next_page <prev_page Set_rate Write !
```

「Defined Logicals」画面には、Performance Monitor に現在アクセス可能なすべての論理名が表示されますが、他のプロセスのテーブルで定義されている論理名は表示できません。

「Defined Logicals」画面では、論理名の値が表示されますが、定義の変更はできません。

「Defined Logicals」画面は、出力ファイルへ書き込まれないので、入力ファイルを使用した再生はできません。

画面上の制約により、論理名は、最初の 28 文字のみが表示されます。

複数バージョンの製品を使用している場合、論理名には、数値の接尾辞は表示されません。たとえば、Oracle Rdb バージョン 6.1 の複数バージョンを使用している場合、「Defined Logicals」画面の論理名には、61 接尾辞は表示されません。◆

2.4 Oracle Trace for OpenVMS

OpenVMS VAX OpenVMS Alpha

Oracle Trace は、Oracle Rdb から収集したイベント・ベースのデータを収集、レポート、表示する製品です。Oracle Trace をシステムにインストールすれば、これを使って、Oracle Trace 向けにインストルメントされている Oracle Rdb やアプリケーションの詳細情報を収集できます。

Oracle Rdb は、Oracle Trace データのログを記録するように、インストルメントされています。**インストルメント**とは、Oracle Trace システム・サービス・ルーチン・コールをアプリケーション・コードに追加することを指します。Oracle Trace システム・サービス・ルーチンを配置することにより、実行時にイベントのデータを収集します。

2.4.1 項（2-75）では、Oracle Rdb でのインストルメントについて説明します。2.4.2 項（2-89）では、イベント・データの選択、収集のスケジューリング、レポート作成、対話方式での表示などの方法について説明します。本書に掲載した Oracle Trace に関する内容に加えて、Oracle Trace のマニュアルも参照してください。

2.4.1 Oracle Rdb のインストルメント

Oracle Trace は、REQUEST_BLR イベントのようなポイント・イベントと、TRANSACTION イベントのような期間イベントという 2 種類のイベントのデータを収集します。

各イベントは、固有の情報（Oracle Trace では項目と呼びます）に対応付けられています。Oracle Trace は、各イベントについて、数多くの種類の項目を収集できます。

表 2-7（2-75）は、Oracle Trace でインストルメントされている Oracle Rdb イベントです。

表 2-7 Oracle Rdb イベント

イベント	説明	イベントのタイプ
DATABASE	データベースへの接続を記録。	ポイント
REQUEST_ACTUAL	ユーザー要求の単一インスタンスの実行を記録。	期間
REQUEST_BLR	要求を実行するたびに、各要求のバイナリ言語表現（Binary Language Representation：BLR）を記録。	ポイント
TRANSACTION	データベース内のトランザクションの開始と終了を記録。	期間

この章で説明する項目は、標準 Oracle Trace リソース使用率項目または Oracle Rdb 固有の項目です。表 2-8（2-76）は、Oracle Trace がデフォルトで収集するリソース使用率項目の一覧です。表 2-8（2-76）の項目は、RESOURCE_ITEMS グループと呼ばれます。必要な情報のみを収集できるように、Oracle Trace では、データ項目を次のようなグループに分類しています。

- RESOURCE_ITEMS
  RESOURCE_ITEMS グループの詳細は、表 2-8（2-76）を参照してください。
- AREA_ITEMS
  AREA_ITEMS グループの詳細は、表 2-10（2-85）を参照してください。

- DATABASE_ITEMS
  DATABASE_ITEMS の詳細は、表 2-11（2-85）を参照してください。
- RDB_CROSS_FAC
  RDB_CROSS_FAC グループの詳細は、表 2-12（2-86）を参照してください。

表 2-8 リソース使用率項目

項目	説明	データ型	使用方法
BIO	バッファ I/O 操作の数	ロングワード	カウンタ
CPU	CPU 時間の合計	ロングワード	カウンタ
CURRENT_PRIO	プロセスの現在の優先順位	ワード	レベル
DIO	ダイレクト I/O 操作の数	ロングワード	カウンタ
PAGEFAULT_IO	ハード・ページ・フォルト（ディスク間のページ・フォルト）の数	ロングワード	カウンタ
PAGEFAULTS	ハードおよびソフト・ページ・フォルトの合計	ロングワード	カウンタ
VIRTUAL_SIZE	プロセスに現在マッピングされている仮想ページの数	ロングワード	レベル
WS_GLOBAL	システム上のプロセス間でグローバルに共有されているワーキング・セットにあるページの数	ロングワード	レベル
WS_PRIVATE	プロセス専用のワーキング・セットにあるページの数	ロングワード	レベル
WS_SIZE	プロセスの現在のワーキング・セットのサイズ	ロングワード	レベル

表 2-9（2-77）は、Oracle Rdb 固有のデータ項目です。

表 2-9 Oracle Rdb データ項目

項目	説明	データ型	使用方法
AIJ_WRITES	データベースの After-image ジャーナル（.aij）ファイルに発行された書込み QIO の数。After-image ジャーナルが有効化されていない場合、この統計は 0。この操作は、After-image レコードを .aij ファイルへ書き込むことにより、ロールフォワード・リカバリ（RMU Recover）を可能にする。	ロングワード	カウンタ
BLR	実行するクエリーのバイナリ言語表現。	ASCII	テキスト 印刷不可
BUFFER_READS	データ（またはスナップショット）ページの論理ページ要求の数。この要求は、Oracle Rdb の PIO サブシステムへのコール。要求されたページがデータベース・ページ・バッファ・プールにキャッシュされていることがあるため、ページ要求によって必ずしも物理的な読取りや書込みが発生するとは限らない。	ロングワード	カウンタ
CLIENT_PC	Oracle Trace へのコールが発生したときの、アプリケーション・プログラムのプログラム・カウンタ（PC）の値。	ロングワード	レベル
COMP_STATUS	OpenVMS の完了ステータス。	ロングワード	レベル
CROSS_FAC_2	サーバー・インデックスの値であり、トランザクションを生成した ACMS プロシージャ・サーバーを一意に示す。	ロングワード	インデックス
CROSS_FAC_3	トランザクションを生成したフォームまたはレポートを一意に示す値。Oracle Rdb 内部インタフェースをコールしない 4GL が使用。	ロングワード	インデックス

表 2-9 Oracle Rdb データ項目（続き）

項目	説明	データ型	使用方法
CROSS_FAC_7	プロシージャ・インデックスの値であり、トランザクションを生成した ACMS プロシージャを一意に示す。	ロングワード	インデックス
CROSS_FAC_14	アプリケーションが使用し、トランザクションを生成したイベントを一意に示す。	ロングワード	インデックス
DB_NAME	データベースの名前。	ASCII	テキスト
DBS_READS	データベース記憶領域とスナップショット・ファイルに発行された読取り QIO の数。この操作では、データベースからデータベース・ページを読み取る。	ロングワード	カウンタ
DBS_WRITES	データベース記憶領域とスナップショット・ファイルに発行された書込み QIO の数。この操作では、変更されたデータベース・ページをデータベースに書き込む。	ロングワード	カウンタ
D_FETCH_RET	データ・ページに読取り権限が要求されたときの、データ・ページの同期要求の数。	ロングワード	カウンタ
D_FETCH_UPD	データ・ページに更新および読取り権限が要求されたときの、データ・ページの同期要求の数。	ロングワード	カウンタ
D_LB_ALLOK	要求されたデータ・ページがユーザーのローカル・バッファ・プールに存在し、必要なページのロックをユーザーが既に取得していた回数。	ロングワード	カウンタ

表 2-9 Oracle Rdb データ項目（続き）

項目	説明	データ型	使用方法
D_LB_GBNEEDLOCK	要求されたデータ・ページがユーザーの割当てセットに存在し、ユーザーが必要なロックを既に取得していた回数。ただし、グローバル・バッファを使用するので、バージョンが正しいことを確認するために追加のロックが必要。バージョンは正しいので、グローバル・バッファを使用するためだけに、追加のロックによるオーバーヘッドが発生。	ロングワード	カウンタ
D_LB_NEEDLOCK	要求されたデータ・ページがユーザーのローカル・バッファ・プールに存在するが、必要なモードでページをロックするために、追加のロックが必要になる回数。	ロングワード	カウンタ
D_LB_OLDVER	要求されたデータ・ページがユーザーのローカル・バッファ・プールに存在するが、ページが古い（ユーザーのローカル・バッファに読み込まれた後、他のユーザーが変更した）ため、ディスクからの読取りが必要になる回数。	ロングワード	カウンタ
D_GB_NEEDLOCK	正しいバージョンのデータ・ページがユーザーの割当てセットに存在するが、必要なモードでページをロックするために、追加のロックが必要になる回数。	ロングワード	カウンタ
D_GB_OLDVER	要求されたデータ・ページがユーザーの割当てセットに存在するが、ページが古い（ユーザーの割当てセットに読み込まれた後、他のユーザーが変更した）ため、ディスクからの読取りが必要になる回数。	ロングワード	カウンタ
D_NOTFOUND_IO	要求されたデータ・ページがバッファ・プールに存在しないので、ディスクからの読取りが必要になる回数。	ロングワード	カウンタ

表 2-9 Oracle Rdb データ項目（続き）

項目	説明	データ型	使用方法
D_NOTFOUND_SYN	要求されたデータ・ページがバッファ・プールに存在しないが、プールの作成にはディスクからの読取りが必要になる回数。	ロングワード	カウンタ
FREE_VM_BYTES	Oracle Rdb がプロセス・ローカル仮想メモリー・プールに返す仮想メモリー・バイト数。GET_VM_BYTES と FREE_VM_BYTES の差は、使用中の仮想メモリーの容量を示す。	ロングワード	カウンタ
GET_VM_BYTES	Oracle Rdb がプロセス・ローカル仮想メモリー・プールから要求した仮想メモリー・バイト数。GET_VM_BYTES と FREE_VM_BYTES の差は、使用中の仮想メモリーの容量を示す。	ロングワード	カウンタ
GLOBAL_TID	分散トランザクションで使用しているトランザクション識別子。	ASCII	プライベート印刷不可
IMAGE_NAME	ディスパッチ・コールを実行しているノードとイメージの名前。	ASCII	テキスト
IO_EXT_BLKCNT	記憶領域ブロック拡張の数。	ロングワード	カウンタ
IO_EXTEND_CNT	記憶領域拡張の数。	ロングワード	カウンタ
IO_EXTEND_STALL	記憶領域拡張ストールの数。	ロングワード	カウンタ
IO_READ_BLKCNT	記憶領域ブロック読取りの数。	ロングワード	カウンタ
IO_READ_CNT	記憶領域読取りの数。	ロングワード	カウンタ
IO_READ_STALL	記憶領域読取りストールの数。	ロングワード	カウンタ
IO_WRITE_BLKCNT	記憶領域ブロック書込みの数。	ロングワード	カウンタ
IO_WRITE_CNT	記憶領域書込みの数。	ロングワード	カウンタ
IO_WRITE_STALL	記憶領域書込みストールの数。	ロングワード	カウンタ
LOCK_MODE	OpenVMS ロック・モード。	ロングワード	レベル

表 2-9 Oracle Rdb データ項目（続き）

項目	説明	データ型	使用方法
LOCK_RELS	既存のロックを解放する $DEQ ロック要求の数。この要求は必ず成功する。REQ_NOT_QUEUED、REQ_DEADLOCKS および LOCK_RELS の合計を LOCK_REQS から差し引いた値が、現在のロックの数。	ロングワード	カウンタ
LOCK_REQS	新しいロックの $ENQ ロック要求の数。要求の成否にかかわらずカウントする。	ロングワード	カウンタ
LOCK_STALL_TIME	トランザクション・スコープ内で、ロックを待っているすべてのユーザーの待ち時間の合計（1/100 秒単位）。この統計は、ロック競合により失われた作業量を相対的に示す。	ロングワード	カウンタ
PROM_DEADLOCKS	ストールした $ENQ ロック要求の数であり、既存のロックを更に高いロック・モードへプロモートした結果、デッドロックが発生したもの。Oracle Rdb は、ほとんどのデッドロックをアプリケーション・プログラムに対して透過的に解決および再試行する。したがって、この数値は、アプリケーション・プログラムに報告されたデッドロックの数を示すわけではない。	ロングワード	カウンタ
REQ_DEADLOCKS	ストールした $ENQ ロック要求の数であり、結果としてデッドロックが発生したもの。Oracle Rdb は、ほとんどのデッドロックをアプリケーション・プログラムに対して透過的に解決および再試行する。したがって、この数値は、アプリケーション・プログラムに報告されたデッドロックの数を示すわけではない。	ロングワード	カウンタ
REQ_ID	データベースの要求の識別子。	ロングワード	レベル

表 2-9 Oracle Rdb データ項目（続き）

項目	説明	データ型	使用方法
REQ_NOT_QUEUED	ロック競合のために即時に拒否された $ENQ ロック要求の数。Oracle Rdb は、待機しないでロックを要求する場合があるが、競合を検出すると、セカンダリ・ロッキング・プロトコルを使用して不要なデッドロックを回避する。この統計は、ロック競合を示す数値の 1 つ。	ロングワード	カウンタ
REQ_STALLS	ロック競合のためにストールした新しいロックの $ENQ ロック要求の数。要求の成否にかかわらずカウントする。この統計は、ロック競合を示す数値の 1 つ。	ロングワード	カウンタ
REQUEST_COUNT	トランザクション内で要求を実行する回数。	ロングワード	レベル
ROOT_READS	データベース・ルート（.rdb）ファイルに発行された読取り QIO の数。Oracle Rdb は、新しいユーザーがデータベースに接続したときと、別の VMScluster ノードでデータベース操作を実行するためにルート・ファイル制御ブロックをリフレッシュする必要があるときに、ルート・ファイルを読み込む。	ロングワード	カウンタ
ROOT_WRITES	データベース・ルート（.rdb）ファイルに発行された書込み QIO の数。ユーザーが COMMIT 文または ROLLBACK 文を発行すると、Oracle Rdb はルート・ファイルを書き込む。他のイベントによっても、ルート・ファイルは更新される。	ロングワード	カウンタ

表 2-9 Oracle Rdb データ項目（続き）

項目	説明	データ型	使用方法
RUJ_READS	データベース Recovery-unit ジャーナル（.ruj）ファイルに発行された読取り QIO の数。.ruj ファイルから Before-image レコードを読み取り、命令やトランザクションをロールバックする。	ロングワード	カウンタ
RUJ_WRITES	データベース Recovery-unit ジャーナル（.ruj）ファイルに発行された書込み QIO の数。命令やトランザクションのロールバックが必要な場合に、.ruj ファイルへ Before-image レコードを書き込む。Before-image を .ruj ファイルに書き込まないと、対応するデータベース・ページをデータベースへ書き戻すことはできない。	ロングワード	カウンタ
STREAM_ID	データベースへの接続を示す識別子。データベースから見たデータベース・ユーザーを示す。	ロングワード	レベル
S_FETCH_RET	SPAM ページに読取り権限が要求されたときの、SPAM ページの同期要求の数。	ロングワード	カウンタ
S_FETCH_UPD	SPAM ページに更新および読取り権限が要求されたときの、SPAM ページの同期要求の数。	ロングワード	カウンタ
S_LB_ALLOK	要求された SPAM ページがユーザーのローカル・バッファ・プールに存在し、必要なページのロックをユーザーが既に取得していた回数。	ロングワード	カウンタ

表 2-9 Oracle Rdb データ項目（続き）

項目	説明	データ型	使用方法
S_LB_GBNEEDLOCK	要求された SPAM ページがユーザーの割当てセットに存在し、ユーザーが必要なロックを既に取得していた回数。ただし、グローバル・バッファを使用するため、バージョンが正しいことを確認するために追加のロックが必要。バージョンは正しいので、グローバル・バッファを使用するためだけに、追加のロックによるオーバーヘッドが発生。	ロングワード	カウンタ
S_LB_NEEDLOCK	要求された SPAM ページがユーザーのローカル・バッファ・プールに存在するが、必要なモードでページをロックするために、追加のロックが必要になる回数。	ロングワード	カウンタ
S_LB_OLDVER	要求された SPAM ページがユーザーのローカル・バッファ・プールに存在するが、ページが古い（ユーザーのローカル・バッファに読み込まれた後、他のユーザーが変更した）ので、ディスクからの読取りが必要になる回数。	ロングワード	カウンタ
S_GB_NEEDLOCK	正しいバージョンの SPAM ページがユーザーの割当てセットに存在するが、必要なモードでページをロックするために、追加のロックが必要になる回数。	ロングワード	カウンタ
S_GB_OLDVER	要求された SPAM ページがユーザーの割当てセットに存在するが、ページが古い（ユーザーの割当てセットに読み込まれた後、他のユーザーが変更した）ため、ディスクからの読取りが必要になる回数。	ロングワード	カウンタ
S_NOTFOUND_IO	要求された SPAM ページがバッファ・プールに存在しないため、ディスクからの読取りが必要になる回数。	ロングワード	カウンタ

表 2-9 Oracle Rdb データ項目（続き）

項目	説明	データ型	使用方法
S_NOTFOUND_SYN	要求されたデータ・ページがバッファ・プールに存在しないが、ディスクからの読取りを実行しなくてもプールの作成が可能になる回数。	ロングワード	カウンタ
TRANS_ID	トランザクションに対応付けられた識別子。	ASCII	プライベート印刷不可

AREA_ITEMS グループは、Oracle Rdb 固有です。表 2-10（2-85）は、AREA_ITEMS グループに対応付けられている項目です。データベース内の記憶領域には、それぞれ 9 つの領域項目があります。データベースに記憶領域が 10 ある場合、領域項目は 90 となります。

表 2-10 AREA_ITEMS グループに対応付けられている項目

IO_EXT_BLKCNT	IO_EXTEND_CNT	IO_EXTEND_STALL
IO_READ_BLKCNT	IO_READ_CNT	IO_READ_STALL
IO_WRITE_BLKCNT	IO_WRITE_CNT	IO_WRITE_STALL

DATABASE_ITEMS グループは、Oracle Rdb 固有です。表 2-11（2-85）は、DATABASE_ITEMS グループに対応付けられている項目です。

表 2-11 DATABASE_ITEMS グループに対応付けられている項目

AIJ_WRITES	BUFFER_READS	DBS_READS
DBS_WRITES	D_FETCH_RET	D_FETCH_UPD
D_LB_ALLOK	D_LB_GBNEEDLOCK	D_LB_NEEDLOCK
D_LB_OLDVER	D_GB_NEEDLOCK	D_GB_OLDVER
D_NOTFOUND_IO	D_NOTFOUND_SYN	S_FETCH_RET
S_FETCH_UPD	S_LB_ALLOK	S_LB_GBNEEDLOCK
S_LB_NEEDLOCK	S_LB_OLDVER	S_GB_NEEDLOCK
S_GB_OLDVER	S_NOTFOUND_IO	S_NOTFOUND_SYN
FREE_VM_BYTES	GET_VM_BYTES	LOCK_RELS
LOCK_REQS	LOCK_STALL_TIME	PROM_DEADLOCKS

表 2-11 DATABASE_ITEMS グループに対応付けられている項目（続き）

REQ_DEADLOCKS	REQ_NOT_QUEUED	REQ_STALLS
ROOT_READS	ROOT_WRITES	RUJ_READS
RUJ_WRITES		

Oracle Trace では、イベントと機能を関連付けることができます。プログラマは、イベントと複数の機能を関連付けることで、イベントが発生する状況を完全に把握できます。プログラマは、2 通りの方法でイベントとアプリケーションを関連付けます。最も一般的なのは、アプリケーション・プログラミング・インタフェース（Application Programming Interface：API）を使って、アプリケーション間でデータ項目を明示的に渡す方法です。ただし、SQL のように、機能が既存の API を変更できない場合や、API を使ってデータ項目を渡せない場合があります。Oracle Trace は、このようなアプリケーションに対応するため、クロス機能を提供しています。プログラマは、表 2-12（2-86）で示す Oracle Rdb 固有のクロス機能やクロス機能項目を使って、イベントと機能を関連付けます。

表 2-12 RDB_CROSS_FAC グループに対応付けられている項目

CROSS_FAC_2	CROSS_FAC_3	CROSS_FAC_7
CROSS_FAC_14		

表 2-13（2-86）から 表 2-17（2-88）は、コレクション・クラスの項目をイベントごとに示したものです。表 2-18（2-88）から表 2-21（2-89）には、各コレクション・クラスで使用できるイベントを示します。

表 2-13（2-86）は、DATABASE イベントに対応付けられている項目です。

表 2-13 DATABASE イベントに対応付けられている項目

CLIENT_PC	DB_NAME	IMAGE_FILE_NAME
STREAM_ID		

表 2-14（2-86）は、REQUEST_ACTUAL イベントに対応付けられている項目です。

表 2-14 REQUEST_ACTUAL イベントに対応付けられている項目

CLIENT_PC	COMP_STATUS	DATABASE_ITEMS[*1]
REQUEST_OPER	REQ_ID	RESOURCE_ITEMS[*2]
STREAM_ID	TRANS_ID	

[*1] DATABASE_ITEMS グループの項目は、表 2-11（2-85）を参照してください。
[*2] RESOURCE_ITEMS グループの項目は、表 2-8（2-76）を参照してください。

表 2-15（2-87）は、REQUEST_BLR イベントに対応付けられている項目です。

表 2-15 REQUEST_BLR イベントに対応付けられている項目

BLR	CLIENT_PC	REQ_ID
STREAM_ID	TRANS_ID	

Oracle Trace では、項目とイベントはクラスに分類されているので、必要な情報のみを収集できます。この章では、次のクラスについて説明します。

- PERFORMANCE
  PERFORMANCE クラスの詳細は、表 2-18（2-88）を参照してください。
- PERFORMANCE_NO_CF
  PERFORMANCE_NO_CF クラスの詳細は、表 2-19（2-88）を参照してください。
- RDBEXPERT
  RDBEXPERT クラスの詳細は、表 2-20（2-89）を参照してください。
- RDBEXPERT_NO_CF
  RDBEXPERT_NO_CF クラスの詳細は、表 2-21（2-89）を参照してください。

表 2-16（2-87）は、PERFORMANCE および RDBEXPERT クラスの TRANSACTION イベントに対応付けられている項目です。

表 2-16 PERFORMANCE および RDBEXPERT クラスの TRANSACTION イベントに対応付けられている項目

AREA_ITEMS[*1]	CLIENT_PC	DATABASE_ITEMS[*2]
GLOBAL_TID	LOCK_MODE	RDB_CROSS_FAC[*3]
RESOURCE_ITEMS[*4]	STREAM_ID	TRANS_ID

[*1] AREA_ITEMS グループの項目は、表 2-10（2-85）を参照してください。
[*2] DATABASE_ITEMS グループの項目は、表 2-11（2-85）を参照してください。
[*3] RDB_CROSS_FAC グループの項目は、表 2-12（2-86）を参照してください。
[*4] RESOURCE_ITEMS グループの項目は、表 2-8（2-76）を参照してください。

表 2-17（2-88）は、PERFORMANCE_NO_CF および RDBEXPERT_NO_CF クラスの TRANSACTION イベントに対応付けられている項目です。

表 2-17 PERFORMANCE_NO_CF および RDBEXPERT_NO_CF クラスの TRANSACTION イベントに対応付けられている項目

CLIENT_PC	DATABASE_ITEMS[*1]	GLOBAL_TID
LOCK_MODE	RESOURCE_ITEMS[*2]	STREAM_ID
TRANS_ID		

[*1] DATABASE_ITEMS グループの項目は、表 2-11（2-85）を参照してください。
[*2] RESOURCE_ITEMS グループの項目は、表 2-8（2-76）を参照してください。

表 2-18（2-88）は、PERFORMANCE コレクション・クラスを構成するイベントと項目です。このイベントと項目は、アプリケーションのパフォーマンスを理解する上で重要です。PERFORMANCE クラスは、Oracle Rdb のデフォルトのコレクション・クラスです。

表 2-18 Oracle Rdb の PERFORMANCE クラスで使用可能なイベントと項目

イベント	イベント型	項目
DATABASE	ポイント	表 2-13（2-86）を参照
REQUEST_ACTUAL	期間	表 2-14（2-86）を参照
TRANSACTION	期間	表 2-16（2-87）を参照

表 2-19（2-88）は、PERFORMANCE_NO_CF コレクション・クラスを構成するイベントと項目です。PERFORMANCE_NO_CF コレクション・クラスは、クライアント・データと Oracle Rdb データの相互参照を行わない場合に使用します。

表 2-19 Oracle Rdb の PERFORMANCE_NO_CF クラスで使用可能なイベントと項目

イベント	イベント型	項目
DATABASE	ポイント	表 2-13（2-86）を参照
REQUEST_ACTUAL	期間	表 2-14（2-86）を参照
TRANSACTION	期間	表 2-17（2-88）を参照

表 2-20（2-89）は、RDBEXPERT コレクション・クラスを構成するイベントと項目です。このイベントと項目は、アプリケーションの作業負荷を理解する上で重要です。

表 2-20 Oracle Rdb の RDBEXPERT クラスで使用可能なイベントと項目

イベント	イベント型	項目
DATABASE	ポイント	表 2-13（2-86）を参照
REQUEST_ACTUAL	期間	表 2-14（2-86）を参照
TRANSACTION	期間	表 2-16（2-87）を参照
REQUEST_BLR	ポイント	表 2-15（2-87）を参照

表 2-21（2-89）は、RDBEXPERT_NO_CF コレクション・クラスを構成するイベントと項目です。RDBEXPERT_NO_CF コレクション・クラスは、クライアント・データと Oracle Rdb データの相互参照を行わない場合に使用します。

表 2-21 Oracle Rdb の RDBEXPERT_NO_CF クラスで使用可能なイベントと項目

イベント	イベント型	項目
DATABASE	ポイント	表 2-13（2-86）を参照
REQUEST_ACTUAL	期間	表 2-14（2-86）を参照
TRANSACTION	期間	表 2-17（2-88）を参照
REQUEST_BLR	ポイント	表 2-15（2-87）を参照

デフォルトでは、Oracle Trace は、機能に定義されているすべてのイベントと項目からデータを収集できます。Oracle Rdb では、ALL コレクション・クラスも使用できます。Oracle Rdb の現行バージョンでは、ALL クラスには、RDBEXPERT クラスと同じイベントと項目が含まれます。ただし、今後のバージョンで追加されるイベントや項目については、ALL クラスには追加されますが、RDBEXPERT クラスには追加されない場合があります。

2.4.2 Oracle Trace の使用方法の概要

Oracle Trace では、次の作業を行うことができます。

- 選択の作成
- 収集の起動
- 収集の終了
- イベント・データの表示
- イベント・データ・レポートのフォーマットと取得

2.4.2.1 項（2-90）から 2.5.3 項（2-95）と、Oracle Trace のマニュアルでは、Oracle Trace の詳細な使用方法を説明します。

2.4.2.1 選択の作成

この項では、選択の作成の概要を説明します。選択の作成の詳細は、Oracle Trace のマニュアルを参照してください。

インストルメントは、非常に全般的な機能であり、パフォーマンス・チューニング、Oracle Expert for Rdb への入力、リソース使用率の評価など、さまざまな用途に利用できるデータを提供します。Oracle Trace は、機能の選択によって、収集対象の限定や必要なイベントおよび項目のみの収集など、各ユーザーのニーズに合わせたカスタマイズができます。機能の選択を使わない場合、Oracle Trace は、Oracle Trace 向けにインストルメントしているシステム上で実行されるすべてのアプリケーションのデータと、対応付けられているデータ項目を収集します。

CREATE SELECTION コマンドを実行すると、次のような機能選択が作成されます。

- 選択名
- データ収集の対象となる機能リスト
- 各機能で収集するデータのクラス

次のコマンドは、機能選択 MY_SELECTION で、デフォルト・データを収集する例です。

```
$ COLLECT CREATE SELECTION MY_SELECTION /FACILITY=RDBVMS
```

複数の Oracle Rdb バージョンがインストールされている場合、Oracle Trace は、デフォルトで、作成日が最も新しい Oracle Rdb を使用します。

2.4.2.2 データ収集のスケジューリング

Oracle Trace で Oracle Rdb のデータ収集を開始するには、SCHEDULE COLLECTION コマンドでデータ収集をスケジューリングしておく必要があります。データ収集のスケジューリングでは、次のような内容を指定します。

- 収集データを格納する 1 つまたは複数の出力ファイル
- 開始時間と終了時間（または継続時間）
- 使用する機能の選択
- 収集の範囲（クラスタ全体またはローカル・ノードのみ）

次の例は、データ収集 MY_COLLECTION を、今日の 11:00 A.M. に開始して正午に終了するスケジューリングを示しています。データ収集は、機能選択 SELECT_ALL を使用し、ローカル・ノード上で実行します。Oracle Trace は、デフォルトのデバイスまたはディレクトリにあるファイル MY_DATA.DAT に収集したデータを格納します。

```
$ COLLECT SCHEDULE COLLECTION MY_COLLECTION MY_DATA.DAT -
_$ /SELECTION=SELECT_ALL -
_$ /BEGINNING=11:00 /ENDING=12:00 -
_$ /NOCLUSTER /COLLECTION_FILES=(PROTECTION=(W:RW))
%EPC-S-SCHED, Data collection MY_COLLECTION is scheduled
```

ENDING 修飾子のかわりに、DURATION 修飾子を使用できます。継続時間は、OpenVMS の相対時間で指定してください。次に例を示します。

```
$ COLLECT SCHEDULE COLLECTION MY_COLLECTION MY_DATA.DAT -
_$ /SELECTION=MY_SELECTION -
_$ /BEGINNING=11:00 /DURATION="1:" -
_$ /NOCLUSTER /COLLECTION_FILES=(PROTECTION=(W:W))
%EPC-S-SCHED, Data collection MY_COLLECTION is scheduled
```

[NO]CLUSTER 修飾子を使用すると、ローカルまたはクラスタ全体の、いずれかのデータ収集をスケジューリングできます。デフォルトでは、SCHEDULE COLLECTION コマンドは、クラスタ内の各ノードでのデータ収集をスケジューリングします。クラスタのサブネットでのデータ収集をスケジューリングするには、データを収集したい各ノードにログインし、/NOCLUSTER を指定して、そのノード上でローカルなデータ収集のスケジューリングを行ってください。スタンドアロン・システムでは、CLUSTER 修飾子は無視されるので注意してください。

Oracle Trace SCHEDULE COLLECTION コマンドで REGISTRATION_ID 修飾子を指定すると、さらに収集対象を限定できます。詳細は、Oracle Trace のマニュアルを参照してください。

2.4.2.3 収集の終了

開始時間と終了時間を指定しない場合は、次のようなコマンド書式でデータ収集を終了できます。

```
$ COLLECT CANCEL COLLECTION MY_COLLECTION /NOCONFIRM
```

また、このコマンドを使って、保留中のデータ収集をキャンセルできます。◆

2.5 Oracle Expert for Rdb での作業負荷情報の収集

OpenVMS VAX OpenVMS Alpha

Oracle Trace は、データベース・アプリケーションから作業負荷データを収集できます。このデータを Oracle Expert for Rdb にインポートすると、データベースの物理設計の最適化に役立てることができます。

作業負荷情報を収集するには、機能選択で、RDBEXPERT または ALL クラスを指定します。Oracle Expert for Rdb 設計プロセスでは、いずれのコレクション・クラスを使用しても差異はありません。次の例では、RDBVMS_WORK という名前の機能選択を作成して、Oracle Rdb から作業負荷データを収集します。

```
$ COLLECT CREATE SELECTION RDBVMS_WORK/OPTIONS -
Option>FACILITY RDBVMS /CLASS=RDBEXPERT /VERSION="V7.0-0"
Option>[Ctrl] + [z]
```

RDBVMS_WORK 選択を使って、データ収集をスケジューリングします。システム上で実行されているすべてのアプリケーションをデータ収集の対象にするのではないときは、REGISTRATION_ID 修飾子で、情報収集する 1 つまたは複数のイメージを指定できます。次の例では、PERSONNEL アプリケーションからのデータ収集をスケジューリングします。

```
$ COLLECT SCHEDULE COLLECTION GET_WORKLOAD WORK.DAT -
_$ /SELECTION=RDBVMS_WORK -
_$ /BEGIN=09:00 /END=12:00 -
_$ /REGISTRATION_ID=($100$DUA1:[TOOLS]PERSONNEL.EXE)
```

データ収集が終了したら、1 つまたは複数のデータ収集ファイルを Oracle Rdb データベースとしてフォーマットします。Oracle Expert for Rdb では、このデータベースをインポートし、作業負荷の設計要素として使用します。データ収集ファイルを RMS ファイルにフォーマットすると、Oracle Expert for Rdb は作業負荷情報をインポートできなくなるので注意してください。次の例では、WORK.DAT のデータを WORKLOAD_DATA.RDB という名前の Oracle Rdb データベースにフォーマットします。

```
$ COLLECT FORMAT WORK.DAT WORKLOAD_DATA
```

Oracle Expert for Rdb を使って、書式付きデータベースから作業負荷データをインポートします。作業負荷データのインポート元として、Oracle Trace の FORMAT コマンドで作成したデータベース・ファイルの名前を指定してください。

詳細は、Oracle Expert for Rdb のマニュアルを参照してください。

2.5.1 イベント・データのインタラクティブ表示

Oracle Trace Monitor では、オープンまたは事前にクローズしたデータ収集ファイルから、イベント・ベースのデータをインタラクティブに表示できます。Monitor には、Motif ベースのウィンドウ・インタフェースがあり、イメージ・レベルからイベント・レベルの情報へ移動できます。

Monitor を起動するには、次の書式でコマンドを実行してください。

```
$ COLLECT MONITOR SAMPLE_DATA.DAT
```

事前にクローズしたデータ収集ファイルからデータを表示したい場合は、MONITOR コマンドの REPLAY 修飾子を使います。Monitor を起動し、事前に収集したデータ収集ファイルのデータを表示するには、次のようにコマンドを実行します。

```
$ COLLECT MONITOR/REPLAY SAMPLE_DATA.DAT
```

「Process」ウィンドウの一番上にある「FILE」メニューの「EXIT」オプションを選択しても、Monitor は終了します。

注意：オラクル社は、Oracle Trace Monitor を使ってイベント・ベースのデータを表示する場合、FLUSH_INTERVAL 修飾子を指定して SCHEDULE COLLECTION コマンドを実行することをお薦めします。FLUSH_INTERVAL 修飾子は、Oracle Trace がすべてのプロセス・バッファをデータ収集ファイルに書き込む頻度を秒単位で指定します。FLUSH_INTERVAL 修飾子を指定することにより、Monitor で時間やデータを正確に表示できます。次の例のように、オラクル社は、1 秒または 2 秒をフラッシュ間隔としてお薦めします。

```
$ COLLECT SCHEDULE COLLECTION SAMPLE_COLLECTION
SAMPLE_DATA.DAT- _$ /SELECTION=SAMPLE_SELECTION - _$ /
DURATION=:30 - _$ /FLUSH_INTERVAL=(00:00:02) - _$ /
NOCLUSTER /COLLECTION_FILES=(PROTECTION=(W:RW))
```

2.5.2 収集データを使用したレポート作成

Oracle Trace では、1 つまたは複数の Oracle Trace コレクションを使って収集したデータを元にレポートを作成できます。次のような手順で、Oracle Trace のレポートを作成します。

1. データ・ファイルのフォーマットとマージ
2. レポート作成

2.5.2.1 データ・ファイルのフォーマットとマージ

Oracle Trace で収集データからレポートを作成するには、FORMAT コマンドを実行して、Oracle Rdb データベースにデータ・ファイルのデータをフォーマットする必要があります[*1]。次のコマンドは、ファイル MY_DATA.DAT のデータをフォーマットし、フォーマットしたデータを FORMATTED_DATA.RDB という名前の Oracle Rdb データベースに格納します。

```
$ COLLECT FORMAT MY_DATA.DAT FORMATTED_DATA
```

FORMAT コマンドは、複数のコレクションから収集したデータを、1 つの書式付きデータベースにまとめることもできます。ただし、コレクションは、同じ機能選択を使ってスケジューリングしておく必要があります。

データ・ファイルのマージには、複数のデータ・ファイルを一度に同じ Oracle Rdb データベースにフォーマットする方法と、既存の書式付きデータベースにデータ・ファイルを追加していく方法の、2 通りがあります。次のコマンドは、複数のコレクションで収集したデータを、WEEK.RDB という名前の新しい書式付きデータベースにマージします。

```
$ COLLECT FORMAT MONDAY.DAT,TUESDAY.DAT,WEDNESDAY.DAT WEEK
```

[*1] カスタマイズしたレポートを作成するユーザー向けに、OpenVMS RMS 書式付きファイルが提供されています。詳細は、Oracle Trace のマニュアルを参照してください。

次のコマンドは、既存の書式付きデータベース WEEK.RDB にデータ・ファイル THURSDAY.DAT の内容を追加します。

```
$ COLLECT FORMAT /MERGE THURSDAY.DAT WEEK
```

フォーマット操作のパフォーマンスを向上する最適化パラメータについては、Oracle Trace のマニュアルの FORMAT コマンドを参照してください。

2.5.2.2 レポート作成

Oracle Trace は、フォーマットした Oracle Rdb データベースに格納されたデータをもとに、表形式のレポートを作成します。表 2-22（2-94）は、作成可能な 3 種類のレポートを示します。

表 2-22 Oracle Trace レポート

タイプ	説明
概要レポート	イベントによるリソース消費を統計的に評価する。パフォーマンス評価や容量プランニングに便利。
詳細レポート	各イベントごとに収集された項目の値を示す。インストルメントするアプリケーションのデバッグに便利。
頻度レポート	コレクション内の間隔ごとに、イベントのオカレンスをカウント。インストルメント・アプリケーションの実行状態のトレースに便利。

最も簡単な方法は、REPORT コマンドを実行し、書式付きデータベースに格納されているすべてのレコードから概要レポートを作成し、現行の SYS$OUTPUT デバイス（使用中の端末など）でレポートを表示する方法です。次に例を示します。

```
$ COLLECT REPORT FORMATTED_DATA.RDB
```

> **注意**：Oracle Trace は、選択したコレクション・クラスの各イベントについて、すべてのオカレンスを収集します。収集対象を個々のイベントに限定することはできません。ただし、EVENTS および ITEMS 修飾子を使用すると、レポート対象となるイベントや項目を絞り込めます。詳細レポートの作成では、イベントや項目を指定しないと膨大なサイズになってしまうため、上記の修飾子の使用をお薦めします。

STATISTICS 修飾子は、Oracle Trace が概要レポートで使用する統計を指定します。これにより、カスタマイズ・レポートを作成できます。指定可能なオプションを次に示します。

- ALL
- COUNT（デフォルト）

- MAXIMUM
- MEAN
- MINIMUM
- STANDARD_DEVIATION
- TOTAL
- 95_PERCENTILE

次の例は、STATISTICS 修飾子の MAXIMUM、MINIMUM および MEAN オプションを使って収集したデータをもとに、概要レポートを作成します。

```
$ COLLECT REPORT FORMATTED_DATA.RDB /FACILITY=RDBVMS -
_$ /TYPE=SUMMARY /STATISTICS=(MAXIMUM,MINIMUM,MEAN) -
_$ /OUTPUT=MY_SUMMARY.TXT
```

デフォルトでは、Oracle Trace は、書式付きデータベースに格納されているすべてのイベントについて、収集されたデータを使ってレポートを作成します。EVENTS 修飾子を使用すると、レポートするイベントを絞り込むことができます。次の例は、Oracle Rdb のイベントのサブセットについて収集したデータをもとに、レポートを作成します。

```
$ COLLECT REPORT FORMATTED_DATA.RDB /OUTPUT=MY_REPORT.TXT -
_$ /FACILITY=RDBVMS -
_$ /EVENTS=(DATABASE,TRANSACTION)
```

2.5.3 カスタマイズ・レポートの作成

指定したイベントと項目に関する情報を提供するカスタマイズ・レポートを作成できます。REPORT コマンドで OPTIONS 修飾子を指定すると、それぞれのイベントや項目の特性を指定できます。例 2-7（2-95）は、イベントごとに異なるレポート・フォーマットを使用してレポートを作成する例です。

例 2-7 レポート・オプションを使用したカスタマイズ・レポートの作成

```
COLLECT REPORT MY_DATABASE -
        /SINCE = 1-JAN-1989 -
        /WIDTH = 80 -
        /TYPE = SUMMARY -
        /LENGTH = 66 -
        /OUTPUT = RDB.REPORT -
        /STATISTICS = ALL -
        /TITLE = "Rdb Reports" -
        /OPTIONS
EVENT REQUEST_ACTUAL -
        /FACILITY = RDBVMS -
        /ITEMS = (CLIENT_PC, CPU) -
        /GROUP_BY = (CLIENT_PC) -
```

```
        /STATISTICS = (MINIMUM, MEAN) -
        /SUBTITLE = "Rdb Request Actual Summary Report"
RESTRICTION STREAM_ID 1
EVENT TRANSACTION -
        /FACILITY = RDBVMS -
        /ITEMS = (CLIENT_PC, PAGEFAULTS, PAGEFAULT_IO) -
        /TYPE = DETAIL -
        /SUBTITLE = "Rdb Transaction Detail Report"
EVENT REQUEST_ACTUAL -
        /FACILITY = RDBVMS -
        /GROUP_BY = (CLIENT_PC) -
        /TYPE = FREQUENCY -
        /INTERVAL = SECONDS -
        /SUBTITLE = "Rdb Request Actual Frequency Report"
RESTRICTION NODE RDB4ME, RDB4U
RESTRICTION COLLECTION RDB_COLL
RESTRICTION EPID 2A8002DF,2A8002C1
RESTRICTION IMAGE PAYROLL, INVENTORY
```

このレポートには、書式付きデータベースである MY_DATABASE のデータが使用されています。最初の REPORT コマンドでは、ページの長さや、レポートに含めるデータの日付など、データ修飾子のデフォルト値をいくつか変更しています。OPTIONS 修飾子は、イベントおよび項目修飾子を指定することにより、主な修飾子を変更したり、レポートに制限を追加したりできます。

REPORT コマンドの最初の部分では、特に指定がなければ、タイムスタンプの日付が 1989 年 1 月 1 日以降のデータ収集ファイルのデータを使って、すべてのサブレポートを作成します。レポートは、幅 80 列、ページの長さは 66 行です。サブレポートは、すべての統計を表示した概要レポートで構成されます。レポートは、カレント・ディレクトリにある RDB.REPORT という名前のファイルに書き込まれます。

例 2-7（2-95）の最初のサブレポートは、REQUEST_ACTUAL イベントについて作成されています。必要な修飾子である機能は、RDBVMS です。対象となるすべての項目の中から、CLIENT_PC と CPU のみが表示されます。このレポートは、CLIENT_PC 項目ごとにグループ化します。つまり、統計のサブセットは、CLIENT_PC の値ごとに表示されます。CLIENT_PC は、実際の Oracle Rdb クエリーのイメージのロケーションです。各項目について、最低値と平均値のみが表示されます。サブレポートの最初のページには、サブタイトルが追加されています。RESTRICTION オプションでは、STREAM_ID 項目の値が 1 のイベントのみを含めるように指定しています。

例 2-7（2-95）の 2 番目のサブレポートは、TRANSACTION イベントについて作成されています。このサブレポートでも、修飾子である機能が必要であり、RDBVMS が指定されています。このイベントでのレポート・タイプは、詳細レポートに変更されています。このイベントでは、CLIENT_PC、PAGEFAULTS、PAGEFAULT_IO の 3 項目のみを表示します。

例 2-7（2-95）の 3 番目のサブレポートは、REQUEST_ACTUAL イベントに関する頻度レポートです。最初のサブレポートと同様に、CLIENT_PC 項目ごとにグループ化します。間隔は秒なので、少なくとも 1 つのイベント・オカレンスが記録されている間、1 秒ごとにカウントが表示されます。このレポートには、いくつかの制限が指定されています。PAYROLL または INVENTORY プログラムを実行するプロセス 2A8002DF および 2A8002C1 について、ノード RDB4ME および RDB4U の RDB_COLL コレクションで収集したデータのみが表示されます。

2.5.4 レポート・パフォーマンスの向上

システム上にデータ・オカレンスが多数存在し、システム対話型 SQL を使用している場合、EQUEST_ACTUAL テーブルに索引を追加すると、レポート生成でのパフォーマンスを向上できます。レポートは、STREAM_ID で限定し、CLIENT_PC でグループ化しているので、索引は、限定項目に入力順で付けた後、グループ化項目に入力順で付けます。次に、SQL 構文を示します。

```
SQL> CREATE INDEX MY_INDEX ON
SQL> EPC$1_241_REQUEST_ACTUAL (STREAM_ID, CLIENT_PC);
SQL> COMMIT;
```

索引を追加すると、この書式付きデータベースに他のデータ収集ファイルをマージするときに、遅延が発生します。他の項目で GROUP_BY を使用する場合、索引は効率的ではありません。次の構文を入力すると、データベースをマージする前に索引を削除できます。

```
SQL> DROP INDEX MY_INDEX;
SQL> COMMIT;
```

書式付きデータベースのレイアウトについては、Oracle Trace のマニュアルを参照してください。

対話型 SQL がインストールされている場合は、EPC$IMAGE テーブルを参照すると、データベースのデータを収集するイメージを把握できます。次の SQL クエリーを入力してください。

```
SQL> SELECT IMAGE NAME FROM EPC$IMAGE.
```

書式付きデータベースの詳細については、付録 B（B-1）と Oracle Trace のマニュアルを参照してください。◆

3

パフォーマンスの要因の分析

データベースを本番に移行すると、ユーザーはタスクが完了するまでの時間の変化を実感することがあります。これは、データベースが適切に設計されており、多くのユーザーに使用されていることを表す場合もありますが、データベースのパフォーマンスが問題になっていることを表す場合もあります。この章では、次のパフォーマンスの要因について説明します。

- 設計上の検討事項
- ディスク I/O 操作の削減
- After-image ジャーナル
- CPU 使用率
- データベースの整合性に関する検討事項
- 制約の最適化
- ロック
- インデックスの検索
- レコードの効率的な挿入

データベース・パフォーマンスの問題には、データベースの構造やデータベースを使用するプログラムなど、多くの要因があります。データベース・パフォーマンスの問題のほとんどは、初めから明らかなわけではありません。多くの場合、時が経つにつれて、また使用量が増えるにつれてパフォーマンスが低下します。パフォーマンスの問題は、データベースの設計不良、非効率的なプログラミング、Oracle Rdb やオペレーティング・システム・パラメータの不適切な設定が原因になる場合もあります。

3.1 データベース設計に関する検討事項

データベース・パフォーマンスを改善するには、データに関する理解を深める必要があります。表 3-1（3-2）に、データに関する質問、そのデータから導出されるファクト、データベースの物理的な設計を左右するパラメータを示します。データに関する知識が深まれば、これらのパラメータやこの章で説明する他のパラメータの値について、それだけ正確な見積りが可能になります。これらのパラメータの最適な見積り値の組合せを計算する方法を明確にすれば、アプリケーションの特性とユーザーの目標や目的が指定された場合に、Oracle Rdb ではそのアプリケーションのパフォーマンスに関する最適なチューニングが可能になります。こうして、アプリケーションが適切に動作するために重要な条件に対して、犠牲になる条件が決定します。

表 3-1（3-2）の初めのいくつかの質問は、論理的な設計に関連します。ロードしたデータ・モデルは、E-R マッピング、重要なトランザクションすべての経路、トランザクション・ボリュームを表す第 3 正規形によって完成されていることが必要です。論理的な設計の詳細は、『Oracle Rdb7 Guide to Database Design and Definition』を参照してください。表 3-1（3-2）の残りの質問は、物理的な設計パラメータのサブセットに関連します。

表 3-1 データに関する知識の確認

データに関する質問	データに関連するファクト	データベース・パラメータの影響
E-R マップは完全か。	エンティティとの関係	インデックス定義、記憶領域定義
データは第 3 正規形に編成されているか。	正規化されたデータ、行サイズ	レコード定義
主要なすべてのトランザクションについてトランザクション分析が済んでいるか。	データベースの使用方法 - 読取り / 更新	クエリーの最適化、領域割当て、ロック
ボリューム分析が済んでいるか。	カーディナリティ、成長	領域割当て、エクステント・オプション
テーブルの数は。	データベースの複雑さ	クエリーの最適化
テーブルをグループ化するのか。	グループ化されたテーブル	複合ページ・フォーマット、ページ・サイズ、クラスタ化、バッファの数、バッファ・サイズ
各テーブルの行サイズは。	レコード・サイズ	領域割当て、ページ・サイズ、断片化

表 3-1 データに関する知識の確認（続き）

データに関する質問	データに関連するファクト	データベース・パラメータの影響
各テーブルの成長率は。	成長	領域割当て、ページ・サイズ、エクステント・オプション、断片化
各テーブルのキーの定義方法は。	主キー、外部キー	インデックス・キーのサイズ、インデックス・ノードのサイズ、充填ファクタ、B ツリー・レベル、ページ・サイズ、I/O、ロック
各ソート・インデックス・キーのサイズは。	インデックス・キーのサイズ	インデックス・ノードのサイズ、充填ファクタ、B ツリー・レベル、ページ・サイズ、I/O、ロック
ソート・インデックスのインデックス・レコードの数とサイズは。	カーディナリティ、レコード・サイズ	インデックス・キーのサイズ、インデックス・ノードのサイズ、充填ファクタ、B ツリー・レベル、ページ・サイズ、I/O、ロック
ハッシュ・インデックスのインデックス・レコードの数とサイズは。	ハッシュ・バケット	インデックス・キーのサイズ、インデックス・ノードのサイズ、ページ・サイズ、I/O
重複を許可されているソート・インデックスはどれか。	親子関係、重複許可	ページ・サイズ、B ツリー・レベル、I/O、ロック
ハッシュ・インデックスは重複を許可 されているか。	親子関係、重複許可	ページ・サイズ、重複ノード・レコード、バッファ・サイズ、バッファの数、バッファ・サイズ、SPAM のしきい値と間隔
重複するソート・インデックス・レコードの数は。	多くの重複レコード	ディープ B ツリーの可能性、ページ・サイズ、I/O、ロック
重複するハッシュ・インデックス・レコードの数は。	多くの重複レコード	多くの重複ノード・レコードの可能性、ページ・サイズ、オーバーフロー、I/O
どのインデックスが一意か。	多くの一意インデックス	シャロー B ツリーの可能性、ページ・サイズ、I/O、ロック
トランザクションあたりの読取りと書込みのボリューム比は。	50/50	ロック、スナップショット、即時 / 遅延の有効化

表 3-1 データに関する知識の確認（続き）

データに関する質問	データに関連するファクト	データベース・パラメータの影響
データベースにアクセスするユーザー数は。	マルチユーザー	ユーザー数、バッファ数、全般的なパフォーマンス
OpenVMS VAX OpenVMS Alpha ◆ VMScluster ノードの数は。	マルチユーザー	全般的なパフォーマンス、バックアップとリカバリ
ほとんど成長しないテーブルはどれか。	不定	圧縮行
更新（読取り / 書込み）のトランザクションはどれか。	不定	非圧縮行、ロック、インデックス・ノード充填ファクタ、断片化、PLACEMENT VIA INDEX 句
クエリー・トランザクションはどれか。	不定	スナップショット、全インデックス・ノード
だれがどのテーブルにどんなクエリーでアクセスするのか。	限定的なアクセス	保護
完全一致のクエリーはどれか。	不定	複合記憶領域、大きいテーブルと小さいテーブル、ハッシュ・インデックス、ソート・インデックス、PLACEMENT VIA INDEX 句
レンジ参照のクエリーはどれか。	不定	均一記憶領域、B ツリー・インデックス
完全一致でレンジ参照のクエリーはどれか。	不定	複合記憶領域と均一記憶領域、ハッシュ・インデックスとソート・インデックス
ショート・トランザクションはどれか。	不定	高速リカバリ
ロング・トランザクションはどれか。	不定	低速リカバリ
トランザクションをリカバリに関して最適化できるか。	不定	リカバリ時間、ダウン時間
ハイボリュームのトランザクションはどれか。	不定	最適化のターゲット

データベースに関する多くの質問があります。その重要度は、何に関する最適化か、物理的な設計の初めてのプロトタイプか既存の設計の改良か、ハードウェアの構成、アプリケーションの特性、データベースに関するクエリーのタイプ（更新と読取りの比率の概数）など、多くの検討事項に依存します。

一部のデータベース・パラメータは他のデータベース・パラメータの計算に使用されます。次に例を示します。

- 各テーブルの行サイズの見積りは初期ページ・サイズを決定するのに使用され、複合ページ・フォーマットの最大行サイズはページ・サイズを決定するのに使用されます。
- テーブルの行の合計数（またはカーディナリティ）と時間の経過に伴うデータベース成長の計画によって、初期領域割当てが決まります。
- ページ・サイズと領域割当ては、SPAM 間隔を決定するのに使用されます。
- 均一ページ・フォーマットの記憶領域では、ページ・サイズに基づいて SPAM 間隔が事前に設定されます。複合ページ・フォーマットの記憶領域では、ページ・サイズと領域割当てに基づいて SPAM 間隔がページの最小数から最大数までの範囲内で変化します。均一ページ・フォーマットと複合ページ・フォーマットの記憶領域について SPAM 間隔を求める計算式は、『Oracle Rdb7 Guide to Database Maintenance』を参照してください。
- 更新集中型の環境では、SPAM の間隔が小さいほど SPAM ページのロックの問題は少なくなりますが、かわりに空き領域を探すのに必要なディスク I/O の数は増加します。間隔が大きい場合は逆になります。
- SPAM のしきい値は、ページ・サイズに関連して格納される行のサイズと頻度に基づいています。
- ページ・サイズとデータベース・テーブルの成長を低く見積りすぎると、断片化が発生する場合があります。

表 3-2（3-6）に、これらのファクタやデータベース・パラメータの一部に関する相互関係を示します。パラメータ値の見積りが不適切であり、次のような相互関係で複雑になり、しかもデータに関する知識が不十分な場合は、アプリケーションを初めてオンラインにしたとき、またはアプリケーションが成熟したときに、パフォーマンスの問題が発生することがあります。

パラメータ値の不適切な見積りによって問題が発生した場合は、SQL の ALTER DATABASE 文か SQL の EXPORT 文と IMPORT 文を使用してデータベースに適した値に設定し直す必要があります。

表 3-2 相互関係のあるデータベース・パフォーマンス・パラメータ

基本パラメータ	関連パラメータ	不適切な見積りによる影響	予想される問題
レコード・サイズ	ページ・サイズ、ページ・フォーマット、SPAM しきい値	断片化、ページの領域不足、むだな領域	小さすぎるページ：断片化、ページの領域不足：スペース使用と検索効率は優れているがストレージ効率は劣る、大きすぎるページ：むだな領域。
		レコードの廃棄	レコードの廃棄はレコードのサイズ指定が小さすぎる場合に発生することがある。
レコードの数	領域割当て	エクステント、正確なサイズ、無駄なディスク領域	小さな領域：エクステントが多すぎる、中程度の領域：エクステントがないか少ない、大きな領域：無駄なディスク領域。
ページ・サイズ	バッファ・サイズ	バッファ・プール	小さすぎる場合：I/O の増大、大きすぎる場合：むだなバッファ領域（メモリーが不足する環境ではページ・フォールトの恐れ）、検索のタイプへの影響（クラスタ・レコードと分散レコード）。
ページ・フォーマット	均一記憶領域	ページ・サイズに基づく固定の SPAM 間隔、SPAM しきい値はテーブルごとに動的に計算されるが論理領域ごとに設定することもできる、ソート・インデックス、ページ・サイズ、B ツリー・レベル、更新によるインデックス・ノードのロック	2 ブロック・ページでは 1089 ページの SPAM 間隔、SPAM しきい値はデフォルトでは動的だが、論理領域ごとに設定することもできる（0 ～ 100%）。

表 3-2 相互関係のあるデータベース・パフォーマンス・パラメータ（続き）

基本パラメータ	関連パラメータ	不適切な見積りによる影響	予想される問題
ページ・フォーマット	複合記憶領域	ページ・サイズに基づく可変の SPAM 間隔、可変の SPAM しきい値、ハッシュ・インデックス、ソート・インデックス、ハッシュ・インデックスではソート・インデックスで発生し得る更新トランザクションによるインデックス・ノードのロック（なし）、ページサイズ	2 ブロック・ページでは 216 ～ 4008 ページの SPAM 間隔、SPAM しきい値は 0 ～ 100%。
グローバルなバッファ有効	NUMBER IS、USER LIMIT、NUMBER OF BUFFERS	バッファ・プール	バッファ・プールが大きすぎると、オペレーティング・システムは仮想のページングを実行する必要がある。バッファ・プールが小さすぎると、Oracle Rdb が実行するデータベース I/O が増大する。
行キャッシング有効	行キャッシュのサイズ	無駄なディスク領域	行キャッシュ・エントリが大きすぎると、むだなディスク領域が発生します。行キャッシュ・エントリが小さすぎると、レコードをキャッシュに格納できない。
パーティション境界	ストレージ・マップ関連の文 - WITH LIMIT OF 句を伴う STORE USING 句	パーティション・レコード・ストレージ	WITH LIMIT OF の値や列データ型の構文が不適切な場合は、すべてのレコードが最初のパーティションに格納されることがある。ALTER STORAGE MAP 文で REORGANIZE オプションを使用すると、レコード記憶域を再調整できる。

上に示す相互関係のある一部のデータベース・パラメータのリストは、計画時のガイドとして役立ちます。この表では、データベースの物理的な設計を実装し、最適なパフォーマンスを得るようにアプリケーションを調整する場合に、データベース・パラメータを正しく見積ることの重要性が強調されています。

3.2 ディスク I/O

この項では、データベース・パフォーマンスを左右するディスクの入力と出力（I/O）ファクタに関する情報の収集方法を説明し、I/O を削減してパフォーマンスを改善するためのヒントを提供します。8 章（8-1）にも、I/O リソースに関する考察、I/O 負荷の分散、ディスク I/O の削減に関する説明があります。

3.2.1 ディスク I/O 情報の収集

3.2.1.1 項（3-8）から 3.2.1.6 項（3-18）では、ディスク I/O 操作の分析に役立つ Performance Monitor のいくつかの画面について説明します。I/O デシジョン・ツリーなどの詳細は、8.1 項（8-2）を参照してください。Performance Monitor の起動方法の概要は、2.2 項（2-15）を参照してください。

3.2.1.1 Performance Monitor の「Summary IO Statistics」画面

Performance Monitor の「Summary IO Statistics」画面には、データベース I/O アクティビティの概要と、トランザクションと Verb の実行レートが表示されます。次に、「Summary IO Statistics」画面の例を示します。

```
Node: TRIXIE          Oracle Rdb V7.0-00 Performance Monitor 16-MAY-1996 14:04:54
Rate: 3.00 Seconds             Summary IO Statistics          Elapsed: 00:46:59.38
Page: 1 of 1              SQL_DISK1:[USER]MF_PERSONNEL.RDB;1          Mode: Online
------------------------------------------------------------------------------
statistic.........      rate.per.second.........      total...     average..
name..............      max.      cur.      avg...     count...     per.trans
transactions               0         0        0.0           2           1.0
verb successes             2         0        1.7         250         125.0
verb failures              0         0        0.2          32          16.0
synch data reads           0         0        0.4          66          33.0
synch data writes          0         0        0.0           0           0.0
asynch data reads          0         0        0.4          66          33.0
asynch data writes         0         0        0.0           0           0.0
RUJ file reads             0         0        0.0           0           0.0
RUJ file writes            0         0        0.0           0           0.0
AIJ file reads             0         0        0.0           0           0.0
AIJ file writes            0         0        0.0           0           0.0
ACE file reads             0         0        0.0           0           0.0
ACE file writes            0         0        0.0           0           0.0
```

```
root file reads            0        0        0.1          20          10.0
root file writes           0        0        0.0           5           2.5
------------------------------------------------------------------------------
Exit Graph Help Menu Options Pause Reset Set_rate Time_plot Unreset Write X_plot
```

「Summary IO Statistics」画面の各フィールドの説明は、Performance Monitor のヘルプを参照してください。

3.2.1.2 Performance Monitor の「IO Stall Time」画面

Performance Monitor の「IO Stall Time」画面には、I/O ストール・アクティビティの概要が表示されます。すべての時間は 1/100 秒単位で表示されます。「IO Stall Time」画面には、「IO Statistics」サブメニューからアクセスします。次の例に、「IO Stall Time」画面を示します。

```
Node: TRIXIE         Oracle Rdb V7.0-00 Performance Monitor 16-MAY-1996 14:08:26
Rate: 3.00 Seconds        IO Stall Time (seconds x100)      Elapsed: 00:50:31.47
Page: 1 of 1    KODD$:[R_ANDERSON.WORK.KUTDIS]MF_PERSONNEL.RDB;1    Mode: Online
------------------------------------------------------------------------------
statistic...........    rate.per.second.............. total....... average......
name................    max..... cur..... avg....... count....... per.trans....
root read time                0        0        0.2          82           0.0
root write time             180        0        0.2          71           0.0
data read time                4        0        3.0        1214           0.6
data write time             100        0        0.6         239           0.1
data extend time              0        0        0.0           0           0.0
RUJ read time                 0        0        0.0           0           0.0
RUJ write time              300        0        0.3         110           0.1
RUJ extend time               0        0        0.4         141           0.1
AIJ read time                 0        0        0.1          52           0.0
AIJ write time             1100        0        8.8        3504           1.7
AIJ hiber time             1100        0        8.9        3569           1.7
AIJ extend time               0        0        0.0           0           0.0
Database bind time            0        0        0.0          21           0.0
------------------------------------------------------------------------------
Exit Graph Help Menu Options Pause Reset Set_rate Time_plot Unreset Write X_plot
```

画面の各フィールドの説明は、Performance Monitor のヘルプを参照してください。

3.2.1.3 Performance Monitor の「Stall Messages」画面

Performance Monitor の「Stall Messages」画面には、データベース・ユーザーのストール・アクティビティの概要が表示されます。ユーザーのプロセスに代わって Oracle Rdb がシステム・サービスを発行するたびに、ユーザーがストールします。たとえば、ユーザーがロックまたは物理ディスクの読取りや書込みを待機する間に、ストールが発生します。

デフォルトでは、「Stall Messages」画面には継続時間がミリ秒の場合を含めてすべてのストールが表示されます。高性能で大量の OLTP（Online Transaction Processing）アプリケーションで高速のディスク・デバイス上に大量の I/O が発生すると、データベース管理者（DBA）は OLTP アプリケーションによる多くの短いミリ秒単位のストールと、データベースを使用する他のアプリケーションによる長い重要なストールを区別できなくなる恐れがあります。

[A] を入力して「Stall Messages」水平メニューの「Alarm」オプションを選択すると、「Stall Messages」画面に表示するストールの最低の継続時間を秒で指定できます。たとえば、5 秒のアラーム間隔を指定すると、「Stall Messages」画面には継続時間が 5 秒以上のストールのみが表示されます。アラーム間隔として 0（ゼロ）を指定すると（デフォルト）、「Stall Messages」画面にはすべてのストールが表示されます。

[B] を入力して水平メニューの「Bell」オプションを選択すると、アラーム・ベル・オプションがアクティブになり、このオプションが選択されます。もう一度 [B] を入力すると、アラーム・ベルが解除され、このオプションの選択が解除されます。

アラーム・ベルがアクティブでも、アラーム・オプションがアクティブでなければベルは鳴りません。

アラームとアラーム・ベルが両方ともアクティブな場合は、画面が更新されて（「Set_rate」オプションで指定）ストールが表示されるたびにアラーム・ベルが鳴ります。アラーム間隔をゼロ（0）秒に設定すると、アラームは解除されます。

ストール・メッセージは、Output 修飾子で作成するバイナリ統計ファイルには保存されません。したがって、RMU Show Statistics コマンドを再実行モードで（Input 修飾子を付けて）実行した場合はこの画面を使用できません。

「Stall Messages」画面には、「Process Information」サブメニューからアクセスします。

次の例に、「Stall Messages」画面を示します。

```
Node: TRIXIE         Oracle Rdb V7.0-00 Performance Monitor 16-MAY-1996 14:34:19
Rate: 3.00 Seconds              Stall Messages             Elapsed: 00:02:50.40
Page: 1 of 1     KODD$:[R_ANDERSON.WORK.AIJ]MF_PERSONNEL.RDB;1      Mode: Online
------------------------------------------------------------------------------
Process.ID Since......   Stall.reason........................... Lock.ID.
2100312E:1 13:16:00.03 - writing pages back to database
21002CAC:1 13:16:00.19 - reading pages 1:1197 to 1:1199
21002B30:1 13:16:00.17 - writing ROOT file
210030B2:1 13:16:00.18 - reading pages 1:2118 to 1:2120
210029B3:1 13:16:00.16 - writing pages back to database
21002AB4:1 13:16:00.11 - reading pages 1:4446 to 1:4448
21002638:1 13:16:00.02 - reading pages 1:4503 to 1:4505
------------------------------------------------------------------------------
Alarm Bell Config Exit Filter Help LockID Menu >next_page <prev_page Set_rate Wr
```

この画面には、データベースのユーザー・プロセスがリストアップされ、Performance Monitor が起動したノード上でユーザーが実行した最新のストールが表示されます。ストール・メッセージは画面の更新間隔でしか抽出されないので、多くのストールが見落とされます。特定のプロセスに関する同じストール・メッセージが継続する場合は、問題を示している可能性があります。プロセスがバグチェック・ダンプを記録する場合に、バグチェック・ダンプ・ファイル名が 53 文字を超えて切り捨てられたときにも表示されます。

「Stall Messages」画面には、現在ストールが進行中のプロセスのみが表示されます。プロセスがストールを終了すると、画面には表示されなくなります。現在ストールが進行中のプロセスは、画面の最上部に集めて表示されます。すなわち、最も継続時間の長いストールのプロセスは、画面の最上部に表示されることになります。新しいストールは、画面の下部に表示されます。したがって、同じノード上のすべてのユーザーは同じストール表示ラインを共有しており、アクティブなストール・プロセスのみがストール画面に表示されます。したがって、比較的多くのストール・プロセスを監視できます。

データベースにストールがない場合は、ストールの表示が空白になります。

RMU Show Statistics コマンドの Time 修飾子に負の数を指定すると、画面を強制的に更新できます。たとえば、修飾子 Time=-10 を指定すると、10/100（1/10）秒ごとに画面が更新されます。ただし、このような間隔を指定すると、多くのシステム・リソースを使用することに注意してください（特に小規模な CPU の場合）。

画面に入らないほど多くのストールが進行中の場合は、現行のストールが表示されていない場合があります。Oracle RMU は、現行のストール・メッセージをできるだけ多く画面に表示しようとします。

RMU Show Statistics コマンドに Stall_Log 修飾子を付けてファイル名を指定すると、ストール・メッセージをファイルにリダイレクトできます。Peformance Monitor 内から、ストール・メッセージのファイルへのロギングを開始または停止することもできます。感嘆符 [!] を入力してツール機能を起動し、「Start/Stop stall logging」オプションを選択します。

ただし、あるプロセスが " 待機あり " のロックを要求し、別のプロセスが非互換モードでロックを保持する場合は、必ず "Waiting for !AD（!AC）" というメッセージが表示されます。このメッセージは、データベースのホット・スポットを示す場合があるので、RMU Show Locks コマンドで調べる必要があります。フォーマット文字列 "!AD" はロックのタイプ（記憶領域、ページ、MEMBIT など）を示し、文字列 "!AC" は要求されたロック・モードを示します（PR、CR、EX など）。

次のリストでは、"Waiting for" メッセージについて説明します。

- Waiting for record or page
  "Waiting for record" メッセージと "Waiting for page" メッセージでは、プロセス ID、時刻、レコードまたはページの dbkey が表示されます。
  "Waiting for record" メッセージの dbkey は、論理 dbkey です。次に例を示します。

```
2040037A:2   16:13:36.78  waiting for record 1:0:-4 (CR)
202003A4:5   16:25:18.51  waiting for record 91:155:-1 (CW)
```

例の 1 行目で、2040037A:2 はプロセス ID、16:13:36.78 は時刻です。"XX:YY:ZZ" のフォーマットは dbkey を表し、通常は " 論理領域番号 : ページ番号 : 行番号 " と解釈します。ただし、正の数の場合にのみ行番号を表します。負の数が表示された場合は、レコード ALG（Adjustable Lock Granularity）ロック・レベルを表します。デフォルトでは、3 つのページ・ロック・レベルがあり、負の数は次のように解釈されます。

- -4 論理領域全体
- -3 通常は 1000 データベース・ページの範囲
- -2 通常は 100 データベース・ページの範囲
- -1 通常は 10 データベース・ページの範囲

たとえば例の 2 行目では、dbkey は 155 ページを含む 10 データベース・ページの範囲の論理領域 91 で発生します。
dbkey が論理領域番号を示し、ユーザーがその論理領域の物理領域名を知りたい場合は、次の手順に従います。

1. RMU Dump コマンドに Lareas 修飾子を指定して発行します。次に例を示します。

```
$ RMU/DUMP/LAREAS=RDB$AIP database-name
$ rmu -dump -larea=RDB¥$AIP db-name
```

2. その番号の付いた論理領域の結果のダンプを検索します。
3. 対応する物理領域番号を書き留めておきます。
4. RMU Dump Header コマンドを発行します。次に例を示します。

```
$ RMU/DUMP/HEADER database-name
$ rmu -dump -header db-name
```

   RMU Dump Header コマンドの出力で上記の物理領域番号を参照すると、物理領域の名前がわかります。

システム・テーブル RDB$RELATIONS、RDB$INDICES、RDBVMS$STORAGE_MAP_AREAS で、列 RDB$STORAGE_ID か RDB$INDEX_ID を参照すると、dbkey が表す Oracle Rdb エンティティ（テーブルまたはインデックス）を識別できます。
dbkey のページ番号フィールドは対応する物理領域のページの番号、行番号はそのページのレコードの番号です。
"Waiting for page" メッセージの dbkey は物理 dbkey です。次に例を示します。

```
202004AA:1   16:14:20.15  waiting for page 1:727 (PW)
```

この dbkey フォーマットは、" 物理領域番号 : ページ番号 " と解釈されます。
dbkey が物理領域番号を示し、ユーザーがその物理領域の物理領域名を知りたい場合は、RMU Dump Header コマンドを発行します。コマンドの出力で物理領域番号を参照すると、物理領域の名前がわかります。

RMU Analyze コマンドを発行して変換テーブルを表示することもできます。次に例を示します。

```
$ RMU/ANALYZE/LAREAS/OPTION=DEBUG/END=1/OUTPUT=LAREA.LIS database-name
$ rmu -analyze -lareas -option=debug -end=1 -output=larea.lis db-name
```

このコマンドを使用すると、データベースのすべての論理領域と物理領域の名前と番号を記載した印刷可能なファイルが作成されます。
前の例での CR（Concurrent Read）、CW（Concurrent Write）、PW（Protected Write）は、要求されたロック・モードとしての同時読取り、同時書込み、保護書込みを意味しています。ロック・モードの詳細は、表 3-9（3-67）を参照してください。

- Waiting for dbkey scope
  このメッセージが表示されるのは、DBKEY SCOPE IS TRANSACTION 句を使用して接続したデータベース・ユーザーが読取り / 書込みトランザクションを実行中（CW モードのデータベース・キー・スコープ・ロックがかかる）に、別のユーザーが DBKEY SCOPE IS ATTACH 句（PR モードのデータベース・キー・スコープ・ロックがかかる）を指定して接続しようとした場合です。この状況では、後のユーザーのプロセスは前のトランザクションが完了するまでストールします。
  SQL の ATTACH 文の DBKEY SCOPE IS 句を使用すると、実行時にデータベース・キー・スコープを指定できます。DBKEY SCOPE IS 句を SQL の CREATE DATABASE 文か SQL の IMPORT 文と併用すると、その文を発行したユーザーのセッションの間のみ設定が有効になります。すなわち、この設定がデータベースのルート・ファイルのパラメータになるわけではありません。

- Waiting for snapshot cursor
  このメッセージは、スナップショットが遅れているときにプロセスが読取り専用トランザクションを開始しようとし、現行の読取り専用トランザクションが存在せず、読取り / 書込みトランザクションがアクティブな場合に表示されます。
  「Waiting for snapshot cursor」は、スナップショットが遅れている場合の正常な状態を表します。最初の読取り専用トランザクションが終了する前に、すべての読取り / 書込みトランザクションが開始されると、待機が終了します。

- Waiting for MEMBIT lock
  個々の Oracle Rdb データベースについて、メンバーシップ・データ構造が保持されます。メンバーシップ・データ構造では、指定された時刻にデータベースにアクセスしているノードを追跡します。データベースのメンバーシップ・データ構造は、特定のノードのユーザー・プロセスがデータベースに最初に接続する場合と、特定のノードのユーザー・プロセスがデータベースの接続を最後に解除する場合に更新されます。
  "Waiting for MEMBIT lock" メッセージは、データベースのメンバーシップ・データ構造が更新中であるためにプロセスがストールされたことを意味します。

- Waiting for Client lock
  クライアント・ロックは、Rdb メタデータ・ロックが使用中であることを示します。クライアントという用語は、Rdb が Rdb ロック・サービスのクライアントであることを示します。メタデータ・ロックは、メタデータ（テーブル、インデックス、列の定義）の

メモリー・コピーとディスク上のメタデータとの一貫性を保証するために使用します。"waiting for client lock" メッセージは、データベース・ユーザーが互換性のないロック・モードを要求していることを表します。たとえば、使用中のテーブルを削除しようとすると、削除操作はメタデータ・オブジェクト（テーブルなど）に関する PROTECTED WRITE ロックを要求します。この要求は、現在このテーブルを使用中の他の操作による既存の PROTECTED READ ロックと互換性がありません。
これらのメタデータ・ロックは、3 つのロングワードで構成されます。ロックは、まずテキスト形式で表示され、次に 16 進表記が続きます。テキスト形式では、印刷できない文字がドット（.）で示されます。
16 進出力の一番左の値は、オブジェクトの ID を示します。ID は、テーブル、ルーチン、モジュール、ストレージ・マップ領域について次の内容を表します。

- テーブルとビューについては、ID は RDB$RELATIONS システム・テーブルの RDB$RELATION_ID 列に記載される一意の値を表します。
- ルーチンについては、ID は RDB$ROUTINES システム・テーブルの RDB$ROUTINE_ID 列に記載される一意の値を表します。
- モジュールについては、ID は RDB$MODULES システム・テーブルの RDB$MODULE_ID 列に記載される一意の値を表します。
- ストレージ・マップ領域については、ID は物理領域 ID を表します。ストレージ・マップ領域に関する "waiting for client lock" メッセージは非常にまれです。これは、Oracle Rdb 5.1 より前のバージョンから変換したデータベースで発生する可能性があります。

次に表示される値は、オブジェクト・タイプを示します。次の表に、オブジェクトとそのタイプの 16 進表記を示します。

表 3-3 オブジェクト・タイプの値

オブジェクト	16 進の値
テーブルまたはビュー	00000004
ルーチン	00000006
モジュール	00000015
ストレージ・マップ領域	0000000E

最後の値は、ロック・タイプを表す 16 進の出力です。55 の値は、クライアント・ロックを示します。
次に、「Stall Messages」画面の "waiting for client lock" メッセージの例を示します。

```
Process.ID Since.....   Stall.reason.......................... Lock.ID.
46001105:2 10:40:46.38 - waiting for client '........' 00000019000000400000055
```

次のリストでは、クライアント・ロックの各要素について説明します。

1. '.....' は印刷できない文字を表します。
2. 00000019 は、一意の識別子の 16 進表記で 19 の値（RDB$RELATION_ID = 25）を示します。
3. 00000004 はオブジェクト・タイプ 4（表）を表します。
4. 00000055 はクライアント・ロックを表します。

ロック ID を指定された参照オブジェクトの名前を決定するには、オブジェクト・タイプに基づいて次のクエリーを使用できます。

```
SQL>select RDB$RELATION_NAME from RDB$RELATIONS where RDB$RELATION_ID = 25;
SQL>select RDB$MODULE_NAME from RDB$MODULES where RDB$MODULE_ID = 12;
SQL>select RDB$ROUTINE_NAME from RDB$ROUTINES where RDB$ROUTINE_ID = 7;
```

画面の各フィールドの説明は、Performance Monitor のヘルプを参照してください。

3.2.1.4 Performance Monitor の「DBKEY Information」画面

Performance Monitor の「DBKEY Information」画面は、アクセスの多いページ上の衝突を識別するのに役立ちます。

「DBKEY Information」画面には、「Process Information」サブメニューからアクセスします。次の例に、「DBKEY Information」画面を示します。

```
Node: TRIXIE          Oracle Rdb V7.0-00 Performance Monitor  4-JUN-1996 16:24:52
Rate: 3.00 Seconds              DBKEY Information            Elapsed: 00:03:24.45
Page: 1 of 1       RDBVMS_USER1:[LOGAN.V70]MF_PERSONNEL.RDB;1       Mode: Online
--------------------------------------------------------------------------------
Process.ID  Data.Page..   Snap.Page..   Spam.Page..   AIP.Page..   ABM.Page..
7E80F841:2  1:648         14:27         5:1           1:483
7E8058F8:1  5:317         14:22         5:218         1:483
7E8052F6:1  1:648         14:20         5:218         1:483
7E80D440:1  5:65          14:18         5:1           1:483
--------------------------------------------------------------------------------
Exit Help Menu >next_page <prev_page Set_rate Write Zoom !
```

この例は、データ・ページ 1:648 と SPAM ページ 5:1、5:218 へのアクセスが多いことを示しています。

「DBKEY Information」画面には、ノード上でデータベースに接続する各プロセスについて、データ・ページ、スナップショット・ページ、SPAM（領域管理）ページ、AIP（領域インベントリ・ページ）、ABM（領域ビットマップ）の各ページ・カテゴリごとに最後に検索した DBKEY が表示されます。

接続されたプロセスからデータ・ページ検索情報の要素として行番号が提供された場合は、行番号が表示されます。それ以外の場合は、領域とページ番号だけが表示されます。スナップショット、SPAM、AIP、ABM ページについては、領域とページ番号だけが表示されます。

3.2.1.5 Performance Monitor の「Active User Stall Messages」画面

Performance Monitor の「Active User Stall Messages」画面は、デッドロックの可能性を示す現行のストールを検出するのに役立ちます。デッドロックは、データベースのホット・スポット（震源地）になります。この画面は、最後にどんなストールが発生したかを調べる場合にも便利です。

「Active User Stall Messages」画面の内容は、Output 修飾子で作成するバイナリ統計ファイルには保存されません。したがって、RMU Show Statistics コマンドを再実行モードで（Input 修飾子を付けて）実行した場合はこの画面を使用できません。

「Active User Stall Messages」画面は、「Process Information」サブメニューから表示します。

次の例に、「Active User Stall Messages」画面を示します。

```
Node: TRIXIE         Oracle Rdb V7.0-00 Performance Monitor 16-MAY-1996 14:35:32
Rate: 3.00 Seconds         Active User Stall Messages       Elapsed: 00:04:02.79
Page: 1 of 1         SQL_DISK1:[RICK.V70]MF_PERSONNEL.RDB;1         Mode: Online
--------------------------------------------------------------------------------
Process.ID Since......   Stall.reason........................... Lock.ID.
2100312E:1     13:16:00.03 - writing pages back to database
21002CAC:1     13:16:00.19 - reading pages 1:1197 to 1:1199
21002B30:1     13:16:00.17 - writing ROOT file
210030B2:1     13:16:00.18 - reading pages 1:2118 to 1:2120
210029B3:1     13:16:00.16 - writing pages back to database
21002AB4:1     13:16:00.11 - reading pages 1:4446 to 1:4448
21002638:1     13:16:00.02 - reading pages 1:4503 to 1:4505
210032B6:1     13:16:00.05 - Bugcheck: DISK1:[RDB]RDSBUGCHK.DMP;1
--------------------------------------------------------------------------------
Config Exit Help LockID Menu >next_page <prev_page Set_rate Write Zoom !
```

「Stall Messages」画面と「Active User Stall Messages」画面は、フォーマットが同じです。しかし、「Active User Stall Messages」画面には現在ストールしているプロセスと過去にストールしたが現在はストールしていないプロセスに関する情報が表示されるのに対して、「Stall Messages」画面には現在ストールしているプロセスしか表示されません。「Active User Stall Messages」画面は、「Stall Messages」画面と同様に、プロセスでバグチェック・ダンプを記述するときに 53 文字を超えるバグチェック・ダンプ・ファイル名が切り捨てられた場合に表示されます。

RMU Show Statistics コマンドの Time 修飾子に負の数を指定すると、画面を強制的に更新できます。たとえば、修飾子 Time=-10 を指定すると、10/100 (1/10) 秒ごとに画面が更新されます。ただし、この間隔を指定すると、多くのシステム・リソースを使用することに注意してください（特に CPU の小さいマシン）。

画面に入らないほど多くのストールが進行中の場合は、現行のストールが表示されていない場合があります。Oracle RMU は、アクティブなストール・メッセージをできるだけ多く画面に表示しようとします。

コマンドに修飾子を付けてファイル名を指定すると、ストール・メッセージをファイルにリダイレクトできます。Peformance Monitor の内部から、ストール・メッセージのファイルへのロギングを開始または停止することもできます。感嘆符 [!] を入力してツール機能を起動し、「Start/Stop stall logging」オプションを選択します。

画面の各フィールドの説明は、Performance Monitor のヘルプを参照してください。

「Active User Stall Messages」画面には、次のように「Stall Messages」画面にないいくつかのメリットがあります。

- プロセスの表示位置は、特定のページ上で固定されているので静的であり、いつでも容易に検出できます。
- プロセスが現在ストールしていない場合も、プロセスの最後のストール・メッセージが表示されます（必要な場合はロック ID も）。これは、ホット・スポットの可能性を識別するのに有効です。

しかし、「Active User Stall Messages」画面には次のデメリットがあります。

- デッドロックの可能性や長期のストールを抽出するのは困難です（不可能ではありません）。この点では、「Stall Messages」画面の方が便利です。
- 現在ストールしているプロセスを、通常のデータベース・アクセスを実行しているプロセス全体から抽出するのは困難です（不可能ではありません）。

表 3-4（3-17）に、ストール・メッセージを表示する 2 つの画面の比較を示します。

表 3-4 「Stall Messages」画面と「Active User Stall Messages」画面の比較

カテゴリ	Stall Messages	Active User Stall Messages
表示されるプロセス	現行ノード上で現在ストールしているプロセスのみ。	現行ノード上でデータベースに接続されているすべてのプロセス。
プロセスの表示位置	動的 --- 位置は他のプロセスとの相対的なストール継続時間を反映する。	静的 - プロセスはデータベースの接続を解除するまで同じ位置に固定される。
表示順序	ストール継続時間の長い順に表示される。	順序は固定されているが、任意に決まったものである。
現行のストールの識別方法	プロセス・ストール・テキストが表示される。	プロセス・ストール・テキスト開始時刻が表示される。

表 3-4 「Stall Messages」画面と「Active User Stall Messages」画面の比較（続き）

カテゴリ	Stall Messages	Active User Stall Messages
完了したストールの識別方法	プロセス・ストール・テキストは表示されない。	プロセス・ストール・テキスト開始時刻が消去される（メッセージ・テキストは表示されたまま）。
表示の継続時間	ストールが進行中の場合にのみストール・メッセージが表示される。	同じプロセスのストールが再開されるまで最後のストール・メッセージがそのまま表示される。

3.2.1.6 Performance Monitor の「Transaction Duration」画面

Performance Monitor の「Transaction Duration」画面には、データベースのトランザクション処理パフォーマンスがリアルタイムで表示されます。「Transaction Duration」画面は、グラフ形式と数値形式で表示できます。デフォルトはグラフ形式です。

グラフ形式の「Transaction Duration」画面には、0 ～ 1 秒で完了したトランザクションの概数、1 ～ 2 秒で完了したトランザクションの概数、以下同様にして最後に 15 秒以上かかって完了したトランザクションの概数に関するサマリー情報が示されます。次の例に、グラフ形式の「Transaction Duration」画面を示します。

```
Node: TRIXIE         Oracle Rdb V7.0-00 Performance Monitor 16-MAY-1996 14:42:52
Rate: 3.00 Seconds           Transaction Duration           Elapsed: 00:11:23.25
Page: 1 of 1       DISK$:[JOE_USER.WORK.AIJ]MF_PERSONNEL.RDB;1      Mode: Online
------------------------------------------------------------------------------
Transaction rate (per second):   current = 0          average = 3.1
Transaction duration (seconds):  average = 0.3        95th pctile = 0.6
Transaction count:               total =      16389   15+++ =          2
            Scaled distribution of transaction lengths (in seconds)
+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+---+
**   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |
**   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |
**   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |
**   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |
**   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |
**   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |
***  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |
***  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |
***  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |
*****|* * |    |    |    |    |    |    |    *    |    |    |    |    |    *   |
+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+---+
0....1....2....3....4....5....6....7....8....9....10...11...12...13...14...15+++
                      (Each "*" represents 702 transactions)
------------------------------------------------------------------------------
Config Exit Help Menu Numbers Options Reset Set_rate Unreset Write !
```

グラフ形式の「Transaction Duration」画面では、1 つのアスタリスク (*) は 1 ～ n 個のトランザクションを表します。ただし、n は画面の下部に指定されたトランザクション数です。この例では、9 ～ 10 秒の領域に表示された 1 つのアスタリスクは、9 ～ 10 秒かかって完了したトランザクションの数が 1 ～ 702 であることを示します。

各トランザクションの持続時間は、SQL の最初の SET TRANSACTION 文から COMMIT 文か ROLLBACK 文まで測定します。この画面には、平均のトランザクション継続時間の他に、トランザクション継続時間のグラフが表示されます。

トランザクションが完了するたびに、累積のヒストグラムの表示に追加されます。95%の応答時間が継続的に更新され、トランザクションの 95%以上が完了する時間が表示されます。この表示によって、システムの応答時間の実態がわかります。

数値形式の「Transaction Duration」画面には、16 の各時間カテゴリについて正確なトランザクション数が表示されます。実行時間の長いトランザクションの正確な数がわかると、これらのトランザクションがアプリケーション全体のパフォーマンスにおいてどれだけ重要かをより正確に判断できます。

グラフ形式の「Transaction Duration」画面で [N] を入力すると、水平メニューの「Numbers」オプションが選択され、「Transaction Duration」画面はグラフ形式から数値形式に変わります。次の例に、前の例で示した「Transaction Duration」画面を数値形式で示します。

```
Node: MYNODE          Oracle Rdb V7.0-00 Performance Monitor 16-MAY-1996 14:43:50
Rate: 1.00 Second            Transaction Duration           Elapsed: 01:27:26.56
Page: 1 of 1       DISK$:[JOE_USER.WORK.AIJ]MF_PERSONNEL.RDB;1      Mode: Online
------------------------------------------------------------------------------
Total transaction count:      16389
Duration   Tx.Count:   %  #Complete:   % #Incomplete:   %
 0-< 1:       16379  99%      16379  99%          10   1% <- average = 0.3
 1-< 2:           7   0%      16386  99%           3   1%
 2-< 3:           0   0%      16386  99%           3   1%
 3-< 4:           0   0%      16386  99%           3   1%
 4-< 5:           0   0%      16386  99%           3   1%
 5-< 6:           0   0%      16386  99%           3   1%
 6-< 7:           0   0%      16386  99%           3   1%
 7-< 8:           0   0%      16386  99%           3   1%
 8-< 9:           0   0%      16386  99%           3   1%
 9-<10:           1   0%      16387  99%           2   1%
10-<11:           0   0%      16387  99%           2   1%
11-<12:           0   0%      16387  99%           2   1%
12-<13:           0   0%      16387  99%           2   1%
13-<14:           0   0%      16387  99%           2   1%
14-<15:           0   0%      16387  99%           2   1%
 15+++:           2   0%      16389 100%           0   0%
------------------------------------------------------------------------------
Config Exit Graph Help Menu Options Reset Set_rate Unreset Write !
```

数値形式では、各時間カテゴリの正確なトランザクション数を表示する以外に、各時間カテゴリで完了したトランザクション数と完了していないトランザクション数が表示されます。

画面の各フィールドの説明は、Performance Monitor のヘルプを参照してください。

RMU Show Statistics コマンドの Output 修飾子を使用して出力ファイルに統計を収集すると、Input 修飾子による「Transaction Duration」画面の再実行ができます。

数値形式の「Transaction Duration」画面で [G] を入力すると、水平メニューの「Graph」オプションが選択され、画面は数値形式からグラフ形式に変わります。

3.2.1.7 ディスク I/O の削減

Oracle Rdb では、ディスク I/O を削減するために以下の機能を提供します。

- グローバルなバッファ
  グローバルなバッファを有効にすると、Oracle Rdb は特定のノード上のすべてのデータベース・ユーザーが共有できるバッファのプールをセットアップします。これで 2 人のユーザーが同じページをメモリーに読み込むことがなくなるので、ディスク I/O が削減されます。グローバルなバッファの使用方法は、4.1.2 項（4-19）を参照してください。

- 行キャッシュ
  行キャッシュを有効にすると、Oracle Rdb はメモリーを割り当ててアクセス頻度の高い行を格納します。該当するページがディスクにフラッシュ・バックされても、行は行キャッシュに残ります。行キャッシュから行を検索すると、ページ・バッファ・プールやディスクから検索するよりもコード・パスが短くなります。行キャッシュの使用方法は、4.1.3 項（4-61）を参照してください。

- 高速コミット処理
  高速コミット処理を有効にすると、Oracle Rdb はコミットされた更新のディスクへの書込みを、ユーザーが指定したしきい値（チェックポイント）に達するまで遅延します。最後のチェックポイント以降に処理されたすべてのトランザクションが同時にディスクに書き込まれるので、トランザクションがコミットされるたびにディスクを更新するより効率的です。トランザクションのたびに RUJ バッファが .ruj ファイルにフラッシュされなくなるので、RUJ I/O も削減されます。高速コミット処理の使用方法は、4.1.5 項（4-89）を参照してください。

大規模なレコード・ストリームが発生するクエリーを実行すると、オプティマイザで中間テーブルを作成して副クエリーの結果を保存しなければならない場合があります。Oracle Rdb では、この結果を後続の結合操作を実行する順にソートして保存します。論理名 RDMS$DEBUG_FLAGS か RDB_DEBUG_FLAGS 構成パラメータを使用すると、オプティマイザで中間テーブルを使用するかどうかを指定できます (詳細は、C.1 項（C-3）を参照)。

Oracle Rdb で認識し、使用する 2 つの論理名か構成パラメータのいずれかまたは両方を定義すると、指定された時間により多くのトランザクションを完了できる場合があります。論理

名は RDMS$BIND_WORK_FILE と RDMS$BIND_WORK_VM、構成パラメータは RDB_BIND_WORK_FILE と RDB_BIND_WORK_VM です。

OpenVMS VAX OpenVMS Alpha

論理名 RDMS$BIND_WORK_FILE と RDMS$BIND_WORK_VM は、OpenVMS システム上でシステム、グループ、プロセスのいずれかのレベルで定義できます。

OpenVMS システム上の Oracle Rdb 作業ファイルは RMS ファイルなので、DCL SET RMS_DEFAULT コマンドで BUFFER_COUNT 修飾子と BLOCK_COUNT 修飾子に適切な値を指定して RMS マルチバッファ・カウントとマルチブロック・カウントを設定し、OpenVMS システム上の Oracle Rdb 一時テーブルのパフォーマンスを改善することもできます。◆

RDMS$BIND_WORK_VM 論理名と RDB_BIND_WORK_VM 構成パラメータを使用すると、マッチング操作に伴うディスク I/O 操作のオーバーヘッドを軽減できます。この論理名または構成パラメータには、プロセスに対してマッチング操作用に割り当てる仮想メモリー（VM）のサイズをバイトで指定できます。この割当てがすべて使用されると、新たなデータ値はディスク上の一時ファイルに書き込まれます (RDMS$BIND_WORK_FILE が未定義の場合は OpenVMS では SYS$LOGIN)。デフォルトは 10,000 バイトです。指定できる値の上限は、システム上で使用可能なメモリーのサイズのみによって決まります。

OpenVMS VAX OpenVMS Alpha

次の例では、RDMS$BIND_WORK_VM 論理名を 20,000 バイトに定義します。

```
$ DEFINE RDMS$BIND_WORK_VM 20000
```

20,000 という数字は、SYSUAF.DAT 内のユーザー・プロセス割当ての値 PGFLQUOTA から導き出されたバイト数です。I/O 操作はページング・ディスクに転送されているので、バイト数が増大するとユーザー・プロセスのページングが増加します。アプリケーションの最大のスループットを提供する値を指定してください。メモリー管理の詳細は、8.2 項（8-38）を、各プロセスに適したワーキング・セット・パラメータ値を確立する方法は 4.4.3 項（4-189）をそれぞれ参照してください。自動的なワーキング・セット・クォータ・パラメータのチューニングには、一般に次の 2 通りがあります。このためのワーキング・セットの初期値設定に関する指針は、OpenVMS のマニュアルを参照してください。

- 常に変化する時分割環境で、作業負荷が大きなワーキング・セットを要求する場合にいつでも迅速に応答するためのチューニング
- 本番環境で、ワーキング・セットの緩やかな成長を要求する安定した変化の少ない応答時間を実現するためのチューニング ◆

論理名 RDMS$BIND_WORK_FILE か RDB_BIND_WORK_FILE 構成パラメータを使用すると、一時作業ファイルのディスク・デバイスとディレクトリを指定できます。

OpenVMS VAX OpenVMS Alpha

OpenVMS システムのデフォルトでは、ユーザーのホーム・ディレクトリに一時ファイルが作成されます。Oracle Rdb は、マッチング操作でこれらのファイルを使用します。次の例では、RDMS$BIND_WORK_FILE 論理名を使用して WORK$DISK:[RDB.WORK] 上に一時作業ファイルを定義しています。

```
$ ! Assign the work area to another disk with read/write access:
$ DEFINE RDMS$BIND_WORK_FILE WORK$DISK:[RDB.WORK]
```

◆

ディスク I/O 操作を削減するもう 1 つの方法は、論理名 RDMS$RUJ または構成パラメータ RDB_RUJ を使用して .ruj ファイルの配置を制御することです。

3.2.2 データの分散化

問題のデータを多くのテーブルに分散すること、すなわち正規化を進めることで、少数のテーブルを中心にクラスタ化されたデータへのアクセスを改善できる場合があります。

概念的なデータベースすなわち論理データベースの設計には、参照テーブル（読取り専用）とアプリケーション・テーブル（読取り / 書込み）の両方が必要です。アプリケーション・テーブルではクラスタ化が発生する可能性がありますが、ユーザーの情報に関する要件をサポートできるように、なんらかの精錬化や変更が必要な場合があります。設計の結果として、90% のデータベース・アクティビティが合計 10 個のテーブルの 2 ～ 3 個で発生する場合は、次の 1 つ以上の処置によってパフォーマンスが改善されることがあります。

- 正規化を若干犠牲にします。
- アクティブでない行を頻繁にアーカイブします。
- インデックス・ストラテジを変更します。
- 制約を適切に使用します。
- 保護方針をより厳密にします。
- より多くのビューを定義します。

データ配分の詳細は、8.1.2.3 項（8-9）を参照してください。制約とビューの定義については、『Oracle Rdb7 Guide to Database Design and Definition』を参照してください。

3.2.3 データ・コンテンツ --- アクティブな行と非アクティブな行

データベースがロードされ、1 つ以上のアプリケーションのサポートを開始すると、データベース内の行の数が増えて妥当な限界や期待値を超えることがあります。データベース管理者は、現行のアプリケーションをサポートするデータのみがデータベース内に存在することを保証する必要があります。アクティブな行と同じデータベースに現行のデータ以外（履歴データ）を保持すると、アクティブな記憶領域に保持してもパフォーマンスが大幅に低下する恐れがあります。この問題には、次のように複数のソリューションを適用できます。最初の方法が最適です。

- 安定データを格納した記憶領域を読取り専用にしてロックの競合を最小化します。
  履歴データや長期間変更されないデータを格納した記憶領域は、読取り専用にするとロックによる競合を軽減できます。これを実現するには、安定データを新しい記憶領域に移動してこれらを読取り専用に設定するか、記憶領域全体を読取り専用にします。

- アクセス頻度に基づいて、履歴データを別のデバイス上の別のテーブル、またはテープにアンロードします。
  別のデバイス上に同等のデータベース構造を定義し、特定の期間履歴データを格納できます。そのデータへのアクセスが必要なくなると、必要に応じて削除するかテープにロードできます。
- データをサマリー行に移し、別のテーブルまたはマイクロフィッシュや CD-ROM などの他のメディアに格納します。
  定期的なサマリーを作成し、同じシステム上のどこか他の場所のサマリー・テーブルにこのデータを格納します。こうして、情報を保持しながら必要なデータ記憶域を削減します。

以上の方法を併用すると、プライマリ・データ・リソースを最適な効率で保持できます。

3.2.4 データベース・ページの非同期プリフェッチ

Oracle Rdb の非同期プリフェッチ機能を使用すると、プロセスが実際にページを要求する前にページをフェッチすることで、ディスクからページを読み込むまでのプロセスの待機時間が短縮されます。非同期プリフェッチ機能が有効な場合、Oracle Rdb は各プロセスを調べてプロセスの将来のアクセス・パターンを予測します。プロセスの連続的なページ要求から順次スキャンを予測できる場合に、Oracle Rdb は順次スキャンの対象となることが予想されるページをフェッチします。このページのプリフェッチ（ページが要求される前にページをフェッチする）は非同期です。すなわち、ページがプリフェッチされているので、Oracle Rdb はディスクからメモリーに読み込まれるまで待機することなく現在処理中のアクティビティを継続します。

非同期プリフェッチ機能はデフォルトで有効です。例 3-1（3-23）に、SQL 構文を使用して非同期プリフェッチ機能を無効にする方法を示します。

例 3-1 非同期プリフェッチ機能の無効化

```
SQL> ALTER DATABASE FILENAME mf_personnel
  1> ASYNC PREFETCH IS DISABLED;
```

非同期プリフェッチ機能を無効にした後で有効に戻す場合は、ALTER DATABASE 文の ASYNC PREFETCH IS ENABLED 句を使用します。

RDM$BIND_APF_ENABLED 論理名または RDB_BIND_APF_ENABLED 構成パラメータを使用しても、プロセスの非同期プリフェッチ機能の無効化や有効化を行うことができます。

OpenVMS VAX OpenVMS Alpha

次の例に、OpenVMS 上で非同期プリフェッチ機能を無効および有効にする方法を示します。

```
$ ! Disable the asynchronous prefetch feature:
$ !
$ DEFINE RDM$BIND_APF_ENABLED 0
$ !
$ ! The RDM$BIND_APF_ENABLED logical name is translated when a
$ ! process attaches to the database.  This logical name can be
```

```
$ ! a system or process logical.
$ !
$ ! To enable the asynchronous prefetch feature after it has been
$ ! disabled for a process, set the RDM$BIND_APF_ENABLED logical name
$ ! to 1:
$ !
$ DEFINE RDM$BIND_APF_ENABLED 1
$ !
$ ! Or, deassign the logical name:
$ DEASSIGN RDM$BIND_APF_ENABLED
```

◆

オラクル社では、論理名と構成パラメータでなく、SQL 構文の使用をお薦めします。SQL の CREATE DATABASE、ALTER DATABASE、IMPORT の各文の ASYNC PREFETCH 句については、『Oracle Rdb7 SQL Reference Manual』を参照してください。

Oracle Rdb がプロセスから順次スキャンを要求されていると判断するページは、**トリガー・ページ**と呼ばれます。Oracle Rdb は、トリガー・ページを検出するとページのプリフェッチを開始します。Oracle Rdb では、この項で後述するように、トリガー・ページが複合フォーマット記憶領域か均一フォーマット記憶領域かによって、異なるプリフェッチ方法をとります。

Oracle Rdb がプロセス用にプリフェッチできるバッファの数は、**バッファの深さ**と呼ばれます。Oracle Rdb では、次の式を使用してデフォルトのバッファの深さが決まります。
デフォルトの深さ = 割当てセット内のバッファ /4 または 8 の小さい方

たとえば、プロセスの割当てセット内に 100 バッファある場合、デフォルトは 25 バッファ (100 バッファ /4=25 バッファ) または 8 バッファです。8 バッファは 25 バッファより小さいので、Oracle Rdb はプロセスの割当てセットの 8 バッファをプリフェッチに使用します。

ASYNC PREFETCH IS ENABLED 句の DEPTH IS パラメータを使用すると、デフォルトの深さを変更できます。例 3-2（3-24）に、プロセスの割当てセットの 10 バッファをプリフェッチに使用するように指定する方法を示します。

例 3-2 プリフェッチするバッファ数の指定

```
SQL> ALTER DATABASE FILE mf_personnel
  1> ASYNC PREFETCH IS ENABLED
  2> (DEPTH IS 10);
SQL>
```

Oracle Rdb は、複合フォーマット記憶領域内にトリガー・ページを検出すると、トリガー・ページの番号に 1 を加えて次のページ（最初にプリフェッチするページ）を決定します。複合フォーマット記憶領域では、Oracle Rdb は 1 回の読取り操作でプロセスのバッファの深さをプリフェッチすることでディスク・アクセスを最適化します。Oracle Rdb は、プロセスのバッファの深さを調べ、別のトリガー・ページを検出するとプリフェッチを継続します。

Oracle Rdb は、均一フォーマット記憶領域内にトリガー・ページを検出すると、トリガー・ページの SPAM ページから論理領域内の次のページ（最初にプリフェッチするページ）を決定します。また、均一記憶領域では、Oracle Rdb はバッファの深さのバッファごとに別の読取り操作でプロセスのバッファの深さを一度に 1 つずつプリフェッチします（これに対して、複合フォーマット記憶領域のプリフェッチではバッファの深さのすべてのバッファを 1 回の読取り操作で読み込みます）。Oracle Rdb は、プロセスのバッファの深さを調べ、別のトリガー・ページを検出すると、プリフェッチを続けます。均一記憶領域内のプリフェッチは SPAM ページ・レベルとデータ・ページ・レベルで発生します。すなわち、Oracle Rdb は領域ビット・マップ（ABM）ページを使用して、どの SPAM ページが特定の論理領域のどのデータベース・ページ・クランプにマップされるかを決定します。

Oracle Rdb は、複合フォーマット記憶領域または均一フォーマット記憶領域のプリフェッチを開始すると、プリフェッチするページを一度に 1 バッファずつ調べます。ただし、プリフェッチするページがメモリー内にあれば、非同期プリフェッチ要求は無視されます。

非同期プリフェッチ機能の利点は、Oracle Rdb がディスクからページを読み込むまでの待機時間に必要なストールの回数を削減できることです。ページの非同期プリフェッチ機能が有効でない場合、Oracle Rdb はディスクからの読取り操作のたびにストールする必要があります。Oracle Rdb が非同期にページをフェッチすればストールは不要になります。したがって、Oracle Rdb はプロセスが必要とするページを要求する前に正しく予想するたびに、通常はページの読取りに伴って発生するストールの時間を節約できます。この結果、様々なアプリケーションのパフォーマンスを大幅に改善できます。

非同期プリフェッチ機能を使用すると、多くの場合、割当てセット 20 バッファ以上のプロセスのパフォーマンスが改善されます。

Performance Monitor の「Asynchronous IO Statistics」画面には、データベースで発生する非同期プリフェッチに関する情報を表示できます。「Asynchronous IO Statistics」画面の詳細は、4.1.1.5 項（4-11）を参照してください。

3.2.5 非同期バッチ書込み操作

Oracle Rdb は、トランザクション実行中にページの読取りと書込みを行います。デフォルトでは、非同期バッチ書込み操作をサポートし、データベース・プロセスがディスクへの書込み完了を待機する間に発生するストールの回数を減少させます。

非同期バッチ書込み操作の目標は、常にプロセスにストールを発生することなくプロセスごとに割当てセットの一部のバッファを使用した書込み操作の進行を可能にし、データベース・パフォーマンスを向上させることです。個々のプロセスでは、プロセスの割当てセットのバッファがすべて変更される前に非同期書込み操作が発行される場合に最高のパフォーマンスが得られます（Oracle Rdb がユーザー・プロセスに割り当てるバッファ数を決定する方法の詳細は、4.1.2.2 項（4-25）を参照）。

非同期バッチ書込み機能を使用すると、次の利点があります。

- サーバーのストールが減少します。
- 各サーバーの効率が向上するので、同等のアプリケーション・スループットを得るために必要なサーバーの台数が少なくなります。
- アプリケーション更新の応答時間が短縮されます。
- RMU Recover、RMU Load、データベース・リカバリ（DBR）プロセスなど、書込み操作を実行する Oracle Rdb ユーティリティによるパフォーマンスが向上します。

この機能を使用すると、多くの場合に割当てセット 20 バッファ以上のプロセスのパフォーマンスが改善されます。

非同期バッチ書込み操作はデフォルトで有効です。例 3-3（3-26）に、SQL の構文を使用して非同期バッチ書込み操作を無効にする方法を示します。

例 3-3 非同期バッチ書込み操作の無効化

```
SQL> ALTER DATABASE FILENAME mf_personnel
  1> ASYNC BATCH WRITES ARE DISABLED;
```

RDM$BIND_ABW_ENABLED 論理名または RDB_BIND_ABW_ENABLED 構成パラメータを使用しても、プロセスの非同期バッチ書込み操作を無効および有効にすることができます。

OpenVMS VAX OpenVMS Alpha

次の例に、OpenVMS 上で RDM$BIND_ABW_ENABLED を使用する方法を示します。

```
$ ! Disable asynchronous batch-write operations:
$ !
$ DEFINE RDM$BIND_ABW_ENABLED 0
$ !
$ ! The RDM$BIND_ABW_ENABLED logical name is translated when a
$ ! process attaches to the database.  This logical name can be
$ ! a system or process logical.
$ !
$ ! To enable asynchronous batch writes after they have been disabled
$ ! for a process, set the RDM$BIND_ABW_ENABLED logical name to 1:
$ !
$ DEFINE RDM$BIND_ABW_ENABLED 1
$ !
$ ! Or, deassign the logical name:
$ DEASSIGN RDM$BIND_ABW_ENABLED
```

◆

オラクル社では、RDM$BIND_ABW_ENABLED 論理名や RDB_BIND_ABW_ENABLED 構成パラメータではなく、SQL 構文の使用をお薦めします。SQL の CREATE DATABASE、ALTER DATABASE、IMPORT の各文の ASYNC BATCH WRITES 句については、『Oracle Rdb7 SQL Reference Manual』を参照してください。

CLEAN BUFFER COUNT パラメータと MAXIMUM BUFFER COUNT パラメータを使用すると、プロセスの非同期バッチ書込み操作を制御できます。CLEAN BUFFER COUNT パラ

メータには、プロセスの置換用バッファの LRU キューの末尾に保持するクリーン・バッファの数を指定します。例 3-4（3-27）では、CLEAN BUFFER COUNT が 4 に定義され、プロセスの置換用バッファの LRU キューとの末尾に 4 つのクリーン・バッファを保持するように指定しています。

例 3-4 クリーン・バッファ保持数の指定

```
SQL> ALTER DATABASE FILE mf_personnel
  1> ASYNC BATCH WRITES ARE ENABLED
  2> (CLEAN BUFFER COUNT IS 4 BUFFERS);
```

MAXIMUM BUFFER COUNT パラメータには、プロセスが非同期に書き込むバッファの数を指定します。例 3-5（3-27）では、MAXIMUM BUFFER COUNT が 6 に定義され、プロセスのクリーン・バッファの数が CLEAN BUFFER COUNT で指定した値より小さい場合は必ず 6 つの変更バッファが非同期で書き込まれるように指定します。

例 3-5 プロセスの非同期で書き込まれるバッファ数の指定

```
SQL> ALTER DATABASE FILE mf_personnel
  1> ASYNC BATCH WRITES ARE ENABLED
  2> (MAXIMUM BUFFER COUNT IS 6 BUFFERS);
```

CLEAN BUFFER COUNT と MAXIMUM BUFFER COUNT が例 3-4（3-27）と例 3-5（3-27）で指定した値をとるプロセスでは、Oracle Rdb はプロセスの置換用バッファの LRU キュー内のクリーン・バッファが 3 つ以下の場合は、必ず 6 つの変更バッファのグループを非同期でディスクに書き込みます。プロセスは 6 つの変更バッファの書込み操作を待機することはありません。非同期バッチ書込み操作が進行中に処理を継続します。

多くの場合、プロセスごとに最高のパフォーマンスを提供する値を決定するには、CLEAN BUFFER COUNT と MAXIMUM BUFFER COUNT に様々な値を指定してみる必要があります。一般に、CLEAN BUFFER COUNT の値を高く設定しすぎない方がパフォーマンスが向上します。この値が高いと、CPU の使用率が高くなることがあります。しかし、非同期バッチ書込み操作を予測し、ストールなしに処理を継続するのに十分なクリーン・バッファが残っている間に Oracle Rdb にこの操作を開始させる必要もあります。したがって、この値が低すぎるのも問題です。CLEAN BUFFER COUNT のデフォルト値は 5 です。

MAXIMUM BUFFER COUNT 値は高く設定しすぎないようにします。高すぎると、プロセスのバッチ書込み操作が非常に大きくなり、他のプロセスの I/O はこのプロセスのバッチ書込み操作が終了するまでキューに置かれる場合があります（特にバッチ書込み操作がすべて同一のディスクを対象とする場合）。

プロセスの MAXIMUM BUFFER COUNT 値が低すぎると、プロセスはディスクのパラレル化による効果を得られません。たとえば、プロセスの MAXIMUM BUFFER COUNT の値が 2 で、アプリケーションによって 5 個のディスクに書込みを行う場合、プロセスは任意のバッチ書込み操作で最大 2 個のディスクのみに非同期に書込みを行います。MAXIMUM BUFFER COUNT の値を大きくすると、1 回の非同期バッチ書込み操作で書き込むディスク

の数を増すことができます。また、プロセスの MAXIMUM BUFFER COUNT 値が低すぎると、プロセスはディスクの最適化による効果を得られません。たとえば、アプリケーションで多くの値を更新する場合、Oracle Rdb は更新された値をディスクに書き込む前にページ番号でソートします。このページによる値のソートによって、ディスク・ヘッドは同じページのすべての更新を一度の操作で書き込むことができます。あるページで一度に 1 つの更新値を書き込み、後で同じページに戻って別の更新値を書き込むのではありません。MAXIMUM BUFFER COUNT の値が低すぎると、ソートする値の数が少なくなるので、すべての更新を同一のページに書き込むには複数回の書込み操作が必要になります。

適切な MAXIMUM BUFFER COUNT の値を指定すると、Oracle Rdb は常にプロセスのいくつかのバッファを着実に書き込むことができます。MAXIMUM BUFFER COUNT のデフォルト値はマークされたすべてのバッファなので、通常は MAXIMUM BUFFER COUNT の構文を使用して Oracle Rdb がプロセスのマークされた比較的少ないバッファを非同期に書き込むように指定します。

非同期バッチ書込み操作によるプロセスではストールは少なくなりますが、Oracle Rdb でバッファの置換が必要なときに非同期バッチ書込み操作が進行中の場合はストールが発生します。この状況は、CLEAN BUFFER COUNT の値を大きくすることで改善できます。非同期バッチ書込み操作が進行中に、プロセスがあるページに関する別のプロセスからのブロッキング AST を受け取った場合にもストールが発生することがあります。ページ・デッドロック処理の一環としてバッファのロック・レベルを下げなければならない場合や、Oracle Rdb でチェックポイント操作を実行する場合にもストールが発生します。

Performance Monitor の「Asynchronous IO Statistics」画面には、データベースで発生する非同期バッチ書込み操作に関する情報を表示できます。「Asynchronous IO Statistics」画面の詳細は、4.1.1.5 項（4-11）を参照してください。

3.3 CPU 使用率

Performance Monitor の「CPU Utilization」画面には、ノード上のデータベース・プロセスごとの現行の CPU 使用率が全プロセッサ使用率に対するパーセンテージで表示されます。

「Process Information」サブメニューで「CPU Utilization」画面を選択します。次に例を示します。

```
+------------ Select Display -----------+
|                                       |
|  A. Stall Messages                    |
|  B. Active User Stall Messages        |
|  C. Process Accounting                |
|  D. Checkpoint Information (Unsorted) |
|  E. Active User Chart                 |
|  F. CPU Utilization (Unsorted)        |
|  G. DBR Activity                      |
|  H. Monitor Log                       |
|  I. Defined Logicals                  |
|  J. Lock Timeout History              |
|  K. Lock Deadlock History             |
|  L. DBKEY Information                 |
|                                       |
+---------------------------------------+
```

[F] を入力すると、「CPU Utilization」画面が選択されます。次に例を示します。

```
Node: TRIXIE         Oracle Rdb V7.0-00 Performance Monitor 16-MAY-1996 14:47:12
Rate: 1.00 Second          CPU Utilization (unsorted)       Elapsed: 00:49:12.10
Page: 1 of 1     KODD$:[R_ANDERSON.WORK.AIJ]MF_PERSONNEL.RDB;1      Mode: Online
-------------------------------------------------------------------------------
Process.ID Process.name... CPU.Util%      10  20  30  40  50  60  70  80  90 100
                                        +---+---+---+---+---+---+---+---+---+---+
2720B4D6:1 RICK2                  9%    +--*|   |   |   |   |   |   |   |   |   +
2720D279:1sRDM_ALS611_1           1%    |   |   |   |   |   |   |   |   |   |   |
2720C687:1sRDM_ABS611_1           0%    |   |   |   |   |   |   |   |   |   |   +
                                        |   |   |   |   |   |   |   |   |   |   |
---------------------------------------+---+---+---+---+---+---+---+---+---+---+
Config Exit Help Menu >next_page <prev_page Set_rate Write Zoom !
```

「CPU Utilization」画面には、ノード上のアクティブな各データベース・プロセスに関する情報が 1 行で表示されます。プロセスがデータベースに繰り返し接続されても、1 行のみの情報が表示されます。

画面の各行には、1 つのデータベース・プロセスごとの CPU 使用率の情報が表示されます。プロセス ID と各プロセスの名前が表示され、そのプロセスの CPU 使用率のパーセンテージが続きます。画面のヒストグラムの部分には、プロセスの使用率のパーセンテージがグラフ形式で表示されます。

ただし、表示されるプロセスの CPU 使用率には、データベース関連の処理の他に、同じプロセスで実行するデータベースに関連のないアクティビティによる CPU 使用率も含まれます。データベースに固有のアクティビティによる正確な CPU 使用率を追跡する場合、オラクル社では Oracle Trace ソフトウェアの利用をお薦めします[*1]。

[*1] OpenVMS システムのみ

ヒストグラムの 100% の列の下に表示されたプラス記号（+）は、プロセスの CPU 使用率が表示セッション内のある時点で 100% に達したことを示します。別の画面に移動してからこの画面に戻ると、後でこのインジケータがリセットされます。

デフォルトでは、画面上のプロセスは不定（未ソート）の順で表示されます。「Config」オプションを選択すると、「CPU Utilization」画面を手動で構成できます。たとえば、[C] を入力すると、次に示す「CPU Utilization Configuration」メニューが表示されます。

```
+- Select CPU Utilization Configuration -+
|                                        |
|  A.  Unsorted                          |
|  B.  Sort by CPU Utilization           |
|                                        |
+----------------------------------------+
```

オプション「A」を選択すると、「CPU Utilization」画面のプロセスが未ソートの順に表示されます。

オプション「B」を選択すると、「Sort by CPU Utilization」画面のプロセスが CPU 使用率の高い順に表示されます。この構成では、表示レートが 2 秒未満の場合は表示が困難なことがあります。プロセスの CPU 使用率は時間が変わるたびに変化します。オラクル社では、ソートされた表示を 2 秒以上の表示レートで使用することをお薦めします。これで、最も CPU 使用率の高いプロセスを最上位とする線形のビューが表示されます。

「CPU Utilization」画面は Output 修飾子を使用しても保存されないので、Input 修飾子を使用しても再実行できません。

3.4 データベースのルート・ファイルに関する情報の収集

データベースのルート・ファイルは、オブジェクトと呼ばれる一連のコンポーネントで構成されます。各オブジェクトは、AIJ や記憶領域の情報など、データベースに関する特定の情報を表します。ルート・ファイル・オブジェクトとオブジェクト指向のオブジェクトを混同しないでください。

モニターでデータベースを開くと、オブジェクトはデータベースのルート・ファイルから読み込まれ、OpenVMS ではデータベースのグローバル・セクションに、Compaq Tru64 UNIX では共有メモリー・パーティションに格納されます。データベースの物理的な変更の間と実行時に、オブジェクトは時折ルート・ファイルからグローバル・セクションまたは共有メモリー・パーティションにリフレッシュされ、変更され、さらにディスクに書き戻されます。たとえば、トランザクション開始時にトランザクション・シーケンス番号（TSN）を割り当てるには、SEQBLK オブジェクトをフェッチされ、変更し、さらにディスクに書き戻します。

Performance Monitor で次の画面を使用すると、データベースのルート・ファイルで実行される I/O 操作を分析できます。

- Summary Object Statistics
  「Summary Object Statistics」画面には、データベースのすべてのルート・ファイルに関する累積情報が表示されます。
- Objects (one stat type)
  「Objects (one stat type)」画面には、特定のオブジェクトの統計を表示できます。メニューから「Objects (one stat type)」オプションを選択すると、ルート・ファイル・オブジェクトを含むサブメニューが表示されます。サブメニューからオブジェクトを選択でき、選択したオブジェクト（カテゴリごとに抽出）の統計が表示されます。
- Objects (one stat field)
  「Objects (one stat field)」画面には、特定の収集カテゴリの統計を表示できます。メニューから「Objects (one stat field)」オプションを選択すると、カテゴリを含むサブメニューが表示されます。サブメニューからカテゴリを選択でき、そのカテゴリ（オブジェクト・タイプごとに抽出）の統計が表示されます。

画面の各フィールドの説明は、Performance Monitor のヘルプを参照してください。

3.5 After-image ジャーナル

本番環境では、After-image ジャーナル機能が有効でない場合にデータベース・アクティビティが発生することはありません。システム障害が発生した場合に、最新の全体バックアップ・ファイルからデータベース・アプリケーションをリストアし、.aij ファイルにロールフォワードし、最新の全体バックアップ操作以降に完了したすべてのトランザクションを獲得できます。この方法では、最新のバックアップ操作以降に完了したすべてのトランザクションをロールフォワードするので、データベースをシステム障害の直前の状態に確実に戻すことができます。ジャーナル・ストラテジは、次にリストアップする要因に基づいて選択します。個々の要因、あるいはそのすべての組合せは、データベースからユーザー要求への応答の是非に大きく影響することがあります。

- データベースに常時アクセスするユーザーの数
  データベースに日常的にアクセスするユーザーが多いか少ないかにかかわらず、ユーザーがデータを入力中にシステム障害が発生するとその情報は失われます。すべてのトランザクションができるだけ短く設計されていれば、最悪の場合でもコミットされていない最新のアクティブなトランザクションを喪失するだけで済みます。ユーザー・プロセス・リカバリは、システム・リカバリの後でないと実行できません。ユーザーに要求されるのは、最後のソース・ドキュメントや POS トランザクションからデータを再入力する作業だけです。トランザクション・スコープが広いと、リカバリの際の遅延という問題が発生する恐れがあります。コミットされていない最新の更新をユーザーが入力してリカバリ・プロセスを実行する必要があるためです。さらに、ユーザーがデータベースの状態を正確に調べる手順が必要になります。この作業にもかなりの時間がかかります。
- トランザクション・ボリューム
  データベースに最大更新サポートによる期間延長の状態が毎日発生する場合は、多くの

更新トランザクションをその期間だけ .aij ファイルに記録することができます。RMU Backup Online コマンドを使用してデータベース自体のバックアップを定期的に作成します。この種のバックアップ操作では、アクティブ・ユーザーに対してデータベースをクローズする必要はありません。ただし、この操作は静止点を待機してから処理されます。データベースに読取り / 書込みトランザクションがあるかどうかにかかわらず、RMU Backup コマンドが発行されたときにバックアップ操作を開始する場合は、Noquiet_Point 修飾子も使用できます。
トランザクションのボリュームによっては、.aij ファイルが巨大化し、大量のディスク・ストレージを要求し、データベース・パフォーマンスが低下する恐れがあります。この場合は、RMU Backup After_Journal コマンドを時折使用します。RMU Backup After_Journal コマンドは、Oracle Rdb に対して .aij ファイルの情報をテープ・ドライブ上か代替ディスク・ドライブ上のファイルにコピーするように指示します。RMU Backup After_Journal コマンドの詳細は、『Oracle Rdb7 Guide to Database Design and Definition』を参照してください。

- データベースを使用できない時間の許容限度
- 他のデータベースからの依存
  他のデバイスまたはノード上のデータベースで、ユーザーのデータベース内のデータを必要とする場合があります。たとえば、給与部門は給与を処理する場合に mf_personnel データベースから従業員データにアクセスします。データベースの可用性がデバイス障害によって中断された場合、またはジャーナル・ファイルが極端に大きいためにシステム障害後のリカバリに時間がかかる場合は、依存プロセスがアイドル状態でなければなりません。したがって、管理情報システムの効率が低下します。

Oracle Rdb では、After-image ジャーナル用に単一の拡張可能なジャーナルか複数の固定サイズ・ジャーナルを使用できます。多くの場合、オラクル社は複数の固定サイズ・ジャーナルの使用をお薦めします。各ジャーナル・タイプのメリットとデメリットの詳細は、『Oracle Rdb7 Guide to Database Maintenance』の After-image ジャーナルの章を参照してください。

8.1.2.2 項（8-6）に、After-image ジャーナルに関連するパフォーマンスの問題をチェックする方法を示します。

3.5.1 AIJ Log Server（ALS）による After-image ジャーナル・ファイルのディスク書込み操作のパフォーマンス改善

中～大規模な更新アクティビティを伴うマルチユーザー・データベースでは、時折 After-image ジャーナル（.aij）・ファイルがボトルネックになることがあります。ALS（AIJ Log Server）と呼ばれる専用のプロセスを使用すると、.aij ファイルのボトルネックを解消できます。

ALS が有効でない場合、.aij ファイルが存在するディスクへの更新バッファのグループ・コミットを実行する前にユーザー・プロセスは独自の更新バッファとデータベースのグローバル・セクションまたは共有メモリー・パーティションの更新バッファのローカル・ロックを

実行する必要があります。データベース内で大規模な更新アクティビティが発生する場合は、このロックの競合によってパフォーマンスが低下することがあります。

上記のような .aij ファイルのボトルネックを解消するために、Oracle Rdb では AIJ Log Server（ALS）を使用してログ・データを .aij ファイルにフラッシュします。ALS を使用すると、すべてのデータベース・ユーザー・プロセスはグローバル・セクションまたは共有メモリー・パーティションのログ・データへのアクセスを中止し、ALS がすべてのユーザー・プロセスに代わってログ・データをディスクにフラッシュします。ALS は専用のプロセスなので、.aij ファイルへの書込みは ALS を使用しない場合よりもはるかに高速です。ALS を使用すると、グローバル・セクションまたは共有メモリー・パーティションのローカル・ロックに関するユーザー・プロセス間の競合もありません。ALS はロックを要求する唯一のプロセスだからです。

データベースの ALS プロセスの使用を開始する前に、まず ALS の起動モードを自動にするか手動にするかを決定する必要があります。例 3-6（3-33）の ALTER DATABASE 文は、データベースの自動 ALS 起動モードを指定しています。Performance Monitor の「Journaling Information」画面には、データベースの ALS 起動モードが表示されます。

例 3-6 ALS プロセスの自動起動モードの指定

```
SQL> -- Specifying the automatic startup mode for the ALS
SQL> ALTER DATABASE FILENAME database-name
  1> JOURNAL ENABLED (LOG SERVER IS AUTOMATIC);
SQL>
```

ALS 起動モードが自動でデータベース・オープン・モードも自動の場合は、ノード上の最初のプロセスがデータベースに接続すると、そのノードで ALS が起動します。ノード上の最後のプロセスが接続を解除した場合、RMU Server After_Journal Stop コマンドが発行された場合、または RMU Close コマンドが発行された場合は ALS が停止します。ALS 起動モードが自動でデータベース・オープン・モードが手動の場合は、RMU Open コマンドでデータベースが明示的にオープンした場合にそのノード上で ALS が起動します。RMU Close コマンドでデータベースが明示的にクローズした場合、または RMU Server After_Journal Stop コマンドが発行された場合は ALS が停止します。

表 3-5（3-33）にオープン・モードが自動と手動のデータベースについて、ALS 起動モードが自動に設定された場合に ALS が起動および終了する様子を示します。

表 3-5 ALS の起動と終了（ALS 起動モードが自動の場合）

データベース・オープン・モード	ALS の起動	ALS の終了
自動	最初のプロセスがデータベースに接続	最後のユーザーがデータベースの接続を解除した場合、RMU Server After_Journal Stop コマンドまたは RMU Close コマンドが発行された場合

表 3-5 ALS の起動と終了（ALS 起動モードが自動の場合）（続き）

データベース・オープン・モード	ALS の起動	ALS の終了
手動	データベースが RMU Open コマンドでオープンした場合	データベースが RMU Close コマンドでクローズした場合、または RMU Server After_Journal Stop コマンドが発行された場合

デフォルトでは ALS 起動モードは手動に設定されます。例 3-7（3-34）の ALTER DATABASE 文は、手動の ALS 起動モードを指定しています。

例 3-7 ALS プロセスの手動起動モードの指定

```
SQL> -- Specifying the manual startup mode for the ALS
SQL> ALTER DATABASE FILENAME database-name
  1> JOURNAL ENABLED (LOG SERVER IS MANUAL);
SQL>
```

データベースの ALS 起動モードが手動に設定されている場合は、例 3-8（3-34）に示すように RMU Server After_Journal Start コマンドを使用してノード上で ALS を起動する必要があります。

例 3-8 ノード上の ALS プロセスの手動による起動

```
$ RMU/SERVER AFTER_JOURNAL START database-name
$ rmu -server after_journal start database-name
```

RMU Server After_Journal Start コマンドは、データベースがオープンしている場合にのみ ALS プロセスを起動します。したがって、ALS 起動モードが手動でデータベース・オープン・モードが自動の場合は、1 つ以上のプロセスがデータベースに接続しているノード上で RMU Server After_Journal Stop コマンドが発行された場合にのみ、ALS が起動することになります。したがって、ALS 起動モードが手動でデータベース・オープン・モードが自動の場合は、1 つ以上のプロセスがデータベースに接続しているノード上で RMU Server After_Journal Stop コマンドが発行された場合にのみ、ALS が起動することになります。すなわち、データベースがクローズしているときに RMU Server After_Journal Start コマンドを発行しても、ALS プロセスは起動しません。

ALS 起動モードが手動の場合は、ALS は RMU Server After_Journal Stop コマンドが発行されたときに停止します。例 3-9（3-35）に、ノード上で RMU Server After_Journal Stop コマンドを使用して手動でデータベースの ALS プロセスを停止する方法を示します。このコマンドは、ユーザーがデータベースにアクセスしているときにも使用できます。RMU Server After_Journal Stop コマンドを使用して ALS を停止する場合、データベースを使用しているアクティブなプロセスはすべて独自の更新バッファと他のユーザーの更新バッファをグループ・コミット操作によって .aij ファイルに書き込む必要があります。RMU Close コマンドを

使用して ALS プロセスを停止することもできます。この場合、データベース・シャットダウンの一環としてデータベース内のプロセスがすべて停止します。

例 3-9 ノード上の ALS プロセスの手動による停止

```
$ RMU/SERVER AFTER_JOURNAL STOP database-name
$ rmu -server after_journal stop database-name
```

ALS プロセスは、Performance Monitor 実行中に起動および終了できます。感嘆符 [!] を入力してツール機能を起動し、「Start/Stop AIJ Log Server」オプションを選択します。

データベースの ALS の起動モードが手動に設定されており、データベースの ALS プロセスが RMU Server After_Journal Start コマンドを使用して手動で起動されている場合、ALS プロセスは RMU Server After_Journal Stop コマンドを使用して手動で停止されるまで有効です（RMU Close コマンドを使用してデータベースがクローズされないと仮定）。ノード上のすべてのデータベース・ユーザーがデータベースの接続を解除しても、ALS は有効です。

表 3-6（3-35）にオープン・モードが自動と手動のデータベースについて、ALS 起動モードが手動に設定された場合に ALS が起動および終了する様子を示します。

表 3-6 ALS 起動モードが手動の場合の ALS の起動と終了

データベース オープン・モード	ALS の起動	ALS の終了
自動	1 つ以上のプロセスがデータベースに接続しており、RMU Server After_Journal Start コマンドが発行された場合	RMU Server After_Journal Stop コマンドか RMU Close コマンドが発行された場合
手動	データベースが RMU Open コマンドでオープンしており、RMU Server After_Journal Start コマンドが発行された場合	RMU Server After_Journal Stop コマンドが発行された場合、またはデータベースが RMU Close コマンドでクローズした場合

現行ノード上でデータベースの ALS プロセスが起動しているかどうかは、RMU Show Users コマンドで判断できます。例 3-10（3-35）で、プロセス RDM_ALS701 は ALS プロセスです。データベースの ALS プロセスが起動していない場合は、RMU Show Users コマンドの表示に RDM_ALS701 プロセスの情報が表示されません。

OpenVMS VAX OpenVMS Alpha

例 3-10 データベースの ALS プロセスが起動しているかどうかの判断

```
$ RMU/SHOW USERS mf_personnel
Oracle Rdb V7.0-00 on node TRIXIE 16-MAY-1996 14:51:26.32
database $111$DUA68:[RICK]MF_PERSONNEL.RDB;1
    - after-image journal file is SQL_DISK1:[RICK.V70_AIJS]AIJ2.AIJ;1
    - 2 active database users
```

```
    - 3D01029D:1 - RICK, RICK - active user
        - image $111$DUA33:[SQL.V070.][SRC.MID]SQL$70.EXE;2
    - 3D00F921:1 - RDM_ALS701_1, RDBVMS - active user
        - image $111$DUJ0:[SYS5.SYSCOMMON.][SYSEXE]RDMALS701.EXE;49
$
```

◆

次の状況では、RMU Show Users コマンドにデータベース名を指定しないと、ノード上にあるすべてのアクティブ・データベースのユーザーに関する情報が表示されます。

- OpenVMS 上で OpenVMS WORLD、SYSPRV、BYPASS 権限がある場合
- Compaq Tru64 UNIX 上で dbsmgr ユーザーまたはスーパーユーザーの場合

データベースの ALS プロセスが起動している場合は、データベースをオープンした各ノードにはそのノードからデータベースにアクセスするユーザー・プロセス専用の ALS プロセスがあります。

ALS プロセスが異常停止しても（RMU Server After_Journal Stop コマンドと RMU Close コマンド以外の方法で、またはデータベースの接続を最後に解除したプロセスによって）、自動的な再起動は行いません。ALS プロセスはデータベース・リカバリ（DBR）プロセスによってリカバリしますが、手動で再起動する必要があります。RMU Server After_Journal Start コマンドは、通常、ALS モードを手動に設定したデータベースでなければ使用できません。しかし、ALS プロセスが異常停止した場合は、ALS の起動モードが手動か自動かにかかわらず、RMU Server After_Journal Start コマンドを使用して ALS プロセスを再起動する必要があります。

OpenVMS VAX OpenVMS Alpha

ALS プロセスのバグチェック・ダンプは、SYS$SYSTEM ディレクトリ内の RDMALSBUG.DMP ファイルに書き込まれます。◆

3.5.2 電子ディスク上の AIJ キャッシュによる ALS プロセス・パフォーマンスの改善

データベースの AIJ ログ・サーバー（ALS）プロセスを有効にすると、中～大規模な更新アクティビティに関するデータベースのパフォーマンスが改善されます。しかし、ALS プロセスが有効なデータベースでも AIJ の書込み操作とコミット操作に伴う非常に短いストールが発生します。このような短いストールをさらに削減するために、電子ディスク上に AIJ キャッシュと呼ばれるファイルを作成し、AIJ 書込み操作用の一時キャッシュとして利用します。

電子ディスクはバッテリ・バックアップ付きのソリッドステート・ディスクであり、安定した磁気ディスクと同様に機能します。電子ディスクは、磁気ディスクをしのぐパフォーマンスを提供します。ALS プロセス用の AIJ キャッシュは磁気ディスク上にも作成できますが、この機能を使用してパフォーマンスを改善するには電子ディスク上に AIJ キャッシュを作成する必要があります。

電子ディスク上の AIJ キャッシュを有効または無効にするには、SQL の ALTER DATABASE 文を使用します（例 3-11（3-37）を参照）。

例 3-11 ALS プロセス用 AIJ キャッシュの有効化と無効化

```
SQL> -- Enabling the AIJ cache for an ALS process
SQL> ALTER DATABASE FILENAME 'SQL_DISK1:[RICK]mf_personnel'
  1> JOURNAL IS ENABLED
  2> (CACHE FILENAME 'SQL_DISK2:[RICK]pers_cache');
SQL> --
SQL> -- Disabling the AIJ cache for an ALS process
SQL> ALTER DATABASE FILENAME mf_personnel
  1> JOURNAL IS ENABLED
  2> (NO CACHE FILENAME);
```

電子ディスク上の ALS プロセス用 AIJ キャッシュの有効化と無効化はオフライン操作です。したがって、ユーザーがデータベースに接続しているときには実行できません。また、データベースの ALS プロセスがアクティブな場合は、AIJ キャッシュの有効化と無効化を実行できません（ALS プロセスがアクティブかどうかは、3.5.1 項（3-32）に示すように RMU Show Users コマンドで確認できます）。

電子ディスク上の ALS プロセス用 AIJ キャッシュが有効な場合は、SQL の SHOW DATABASE 文を使用すると Performance Monitor の「Journaling Information」画面にキャッシュ・ファイルの名前が表示されます。

クラスタを構成するノードごとに、電子ディスク上の ALS プロセス用 AIJ キャッシュが 64 ブロックに分割されます。クラスタを構成する個々のノード上の ALS プロセスは、同じ AIJ キャッシュの別々の部分を使用します。

電子ディスク上の ALS プロセス用 AIJ キャッシュを使用すると、データベース・アプリケーションの応答時間が短縮され、スループットが向上します。このために必要なスペースはほんのわずかです。AIJ キャッシュはクラスタ内で機能し、バックアップは必要ありません。ALS に障害が発生した場合は、自動データベース・リカバリ（DBR）プロセスの一環として自動的にリカバリします。

3.5.3 WORM 記憶領域の After-image ジャーナルの無効化によるパフォーマンスの改善

Oracle Rdb では、ユーザーが WORM（Write-Once, Read-Many）デバイス上の記憶領域にリスト（セグメント化された文字列）データを格納できます。デフォルトでは、データベースの After-image ジャーナルが有効になると、データが WORM 領域（WORM デバイス上の記憶領域）に書き込まれます。Oracle Rdb では、この情報のログが .aij ファイルにも記録されます。WORM 領域に書き込んだ情報は絶対に上書きされることがないので、WORM の

変更のログを .aij ファイルに記録するのは不要な作業だと考えるユーザーもいます。この場合は、WORM の変更のロギングを無効にすることもできます（特に WORM 領域にデータをロードするとき）。

WORM の変更の .aij ファイルへのロギングを無効にすると、障害発生後にデータベースをロールフォワードするための時間も短縮されます。.aij ファイル内の WORM 領域のジャーナル・レコードは、WORM 領域内の情報が上書きされることがなくても適用する必要があります。この結果、障害発生後のデータベース・リカバリにかかる時間が長くなります。

WORM 領域への書込み操作の AIJ ロギングを無効にするには、SQL の ALTER DATABASE 文か CREATE DATABASE 文を使用します。例 3-12（3-38）に示す ALTER DATABASE 文は、マーケティング・データベースで WORM 領域 1986_EVENTS への書込み操作の AIJ ロギングを無効にしています。これはオフライン操作です（ユーザーがデータベースに接続しているときには実行できません）。

例 3-12 WORM 領域への書込み操作の AIJ ロギングの無効化

```
SQL> ALTER DATABASE FILENAME marketing
  1> ALTER STORAGE AREA 1986_EVENTS
  2> WRITE ONCE (JOURNAL IS DISABLED);
```

注意：WORM 領域への書込み操作の AIJ ロギングを無効にすると、WORM 領域を使用するアプリケーションのパフォーマンスが改善されます。しかし、WORM 領域に書き込んだデータのログが .aij ファイルに記録されないので、WORM メディアに障害が発生した場合にリカバリできるという保証はありません。WORM 領域への書込み操作の AIJ ロギングを無効にしてパフォーマンスを改善する前に、その結果として WORM メディアに障害が発生した場合に WORM 領域内のデータが失われる可能性があることについて慎重に検討する必要があります。

たとえば、WORM 領域への書込み操作の AIJ ロギングが無効にされているとします。WORM 領域に 120 ページが割り当てられており、その中の 100 ページに書き込まれる場合で、50 ページ分の情報が存在する時点で、この領域の最新のバックアップを実行したとします。WORM デバイスに障害が発生すると、最新のバックアップから新しい WORM メディアに WORM 領域をリストアする必要があります。バックアップ済みの最初の 50 ページはリストアされます。データベースには WORM 領域の 50 ～ 120 ページへのポインタはありますが、この分のデータは失われています。このデータベースでは、リストアされなかった WORM 領域内のページのフェッチが要求されるたびに例外が発生します。

また、WORM 領域への書込み操作の AIJ ロギングを、複数回にわたって有効化および無効化したため、正しくバックアップされなかった場合を想定してください。WORM デバイスに障害が発生すると、リストアとロールフォワード操作が完了しても、WORM 領域にはログのない部分に対応する複数の空きが存在します。このような空きへの参照が行われるたびに、例外が発生します。

WORM 領域に関する AIJ ロギングを無効にすると、パフォーマンスが改善されるかわりにデータの信頼性が失われます（WORM デバイスに障害が発生した場合）。WORM 領域への書込み操作の AIJ ロギングを無効にしても、次の処理を実行すると WORM 領域内のデータの喪失を防ぐことができます。

- WORM デバイスを他のデバイスにシャドウします。
- WORM 記憶領域を定期的にバックアップします。
- デバイスに書き込んだマルチメディア・オブジェクトを、バックアップが実行されるまで他のメディアで保持します（たとえば、イメージをスキャンして WORM 領域に格納した場合は、バックアップが実行されるまでオリジナル・イメージを保存します）。

3.6 RMU Optimize After_Journal コマンドのパフォーマンスの改善

RDM$BIND_OPTIMIZE_AIJ_RECLEN 論理名か RDB_BIND_OPTIMIZE_AIJ_RECLEN 構成パラメータを定義すると、RMU Optimize After_Journal コマンドのパフォーマンスを改善できます。

RMU Optimize After_Journal コマンドに Log 修飾子を指定して発行すると、次のようなメッセージが表示されます。

```
%RMU-I-OPTRECLEN, AIJ optimization record length was XXX characters in length
```

XXX は、構成可能な最大のレコード・サイズよりも小さい最大の After-image ジャーナル最適化レコードを示します。構成可能な最大レコード・サイズのデフォルトは 1536 文字です。最適化レコード長は、512 ～ 4096 文字です。

OpenVMS VAX OpenVMS Alpha

次の OpenVMS の例について考察します。

```
$ RMU/OPTIMIZE/AFTER/LOG backup1.aij opt.aij
%RMU-I-LOGOPNAIJ, opened journal file DISK$:[USER.WORK.AIJ]BACKUP1.AIJ;1
%RMU-I-LOGCREOPT, created optimized after-image journal file
DISK$:[USER.WORK.AIJ]OPT.AIJ;40
%RMU-I-OPTRECLEN, AIJ optimization record length was 474 characters in length
%RMU-S-AIJOPTSUC, AIJ optimization completed successfully
%RMU-I-LOGSUMMARY, total 8203 transactions committed
%RMU-I-LOGSUMMARY, total 0 transactions rolled back
```

◆

RDM$BIND_OPTIMIZE_AIJ_RECLEN 論理名か RDB_BIND_OPTIMIZE_AIJ_RECLEN 構成パラメータの値に関するメッセージに表示される After-image ジャーナル最適化レコード長の値を使用して、パフォーマンスを最適化します（この例では 474）。最適化操作の前に適切な値を予測することはできませんが、多くの場合、指定されたアプリケーションに見あう

.aij レコード・サイズが生成されます。前回の最適化操作の値は、多くの場合、次回にも適用できます。選択した値が小さすぎる場合はメッセージが表示されます。

一般に、オラクル社では次の指針に従って After-image ジャーナル最適化の全体的なパフォーマンスを改善することをお薦めします。

- 常に RDMS$BIND_SORT_WORKFILES 論理名か RDB_BIND_SORT_WORKFILES 構成パラメータを使用して、使用する作業ファイルの数を指定します。
- 常に SORTWORKn 論理名を使用して、使用されていないできるだけ高速なデバイス上の一時作業ファイルのファイル名を指定します。
- 同じデバイス上には 2 つ以上のファイルを置かないでください。
- 作業ファイル・デバイスに大きな空き領域がある場合は、使用する作業ファイルの数を少なくします。空き領域が限られている場合は、多くの作業ファイルを使用します。
- 出力を追跡する必要がない場合は、Trace 修飾子を使用しないでください。トレース出力を実行すると、バッファ I/O やダイレクト I/O の数が増大し、経過時間全体が長くなります。
- Log 修飾子を使用すると、OPTRECLEN に関するメッセージが表示されます。
- OPTRECLEN のメッセージの出力 +10% を RDM$BIND_OPTIMIZE_AIJ_RECLEN 論理名か RDB_BIND_OPTIMIZE_AIJ_RECLEN 構成パラメータの値として使用します。最小値は 512、最大値は 4096 です。
- RDM$BIND_OPTIMIZE_AIJ_RECLEN の値よりも大きな DDL レコードや .aij レコードの多い .aij ファイルは最適化しないでください。この種のレコードには即時ソートとフラッシュ操作が必要です。この操作を実行すると、莫大な費用がかかり、大きな出力ファイルが生成されます。レコードの正確な数は、アプリケーションと入力 After-image ジャーナル全体のサイズによって大きく変わります。

DDL レコードと After-image ジャーナル・レコードのサイズが上限を超えても、最適化 .aij 出力ファイルの生成は阻止されません。最適化操作の最高のパフォーマンスが低下するだけです。

ソートおよびフラッシュ操作を強制的に実行する aij レコードに達すると、次のようなメッセージが表示されます。

```
%RMU-I-OPTEXCMAX, TSN XXX record size YYY exceeds maximum ZZZ record size
```

.aij レコードにソートおよびフラッシュ操作を強制的に実行する DDL 操作がある場合は、次のようなメッセージが表示されます。

```
%RMU-I-OPTDDLREC, TSN XXX contains DDL information that cannot be optimized
```

Trace 修飾子を指定した場合にも、内部ソート操作が発生するたびにソート操作に関する一連の統計情報が表示されます。次に例を示します。

```
%RMU-I-OPTSRTSTAT, Number of input records: 10048
%RMU-I-OPTSRTSTAT, Number of sorted records: 10048
%RMU-I-OPTSRTSTAT, Number of output records: 0
%RMU-I-OPTSRTSTAT, Number of merge passes: 1
%RMU-I-OPTSRTSTAT, Number of sort nodes: 6490
%RMU-I-OPTSRTSTAT, Workfile allocation blocks: 29675
```

RDM$BIND_OPTIMIZE_AIJ_RECLEN 論理名か RDB_BIND_OPTIMIZE_AIJ_RECLEN 構成パラメータの重要性を強調するために、次に示す 2 つの同じテストの相違について考察します。

RDM$BIND_OPTIMIZE_AIJ_RECLEN=4096 を使用した場合

```
Direct I/O count :                7816
Elapsed CPU time :       0 00:01:29.91
Connect time :           0 00:04:17.31
```

RDM$BIND_OPTIMIZE_AIJ_RECLEN=512 を使用した場合

```
Direct I/O count :                 925
Elapsed CPU time :       0 00:00:37.70
Connect time :           0 00:01:25.59
```

.aij 最適化レコード長が 4096 の場合は、RDM$BIND_OPTIMIZE_AIJ_RECLEN 論理名か RDB_BIND_OPTIMIZE_AIJ_RECLEN 構成パラメータを 4096 に設定します。ただし、設定する値が大きすぎると、システム・パフォーマンスが低下することがあります。

3.7 制約の最適化

Oracle Rdb では、次のようにいくつかの制約評価操作が最適化されます。

- 存在
- 一意
- 変更
- データベース・キー（dbkey）による検索と消去

次の項では、上記の最適化の特徴について説明します。制約とパフォーマンスの詳細は、8.1.3.1 項（8-23）も参照してください。

3.7.1 存在の制約

存在の制約は、制約内で名前を指定されたテーブルの行を格納または削除をする場合には評価されません。次に例を示します。

```
SQL> ALTER TABLE SALARY_HISTORY
  1> ADD CONSTRAINT CONSTRAINT SH_EMP_ID_EXISTS
```

```
  2> CHECK (EMPLOYEE_ID = ANY (SELECT EMPLOYEE_ID
  3> FROM EMPLOYEES))
  4> DEFERRABLE;
```

Oracle Rdb では、EMPLOYEES テーブルに行を格納する場合や SALARY_HISTORY テーブルから行を削除する場合にはこの制約を評価する必要がありません。

3.7.2 一意の制約

一意の制約は、制約内で名前を指定されたテーブルの行がデータベースから削除される場合には評価されません。次に例を示します。

```
SQL> ALTER TABLE EMPLOYEES
  1> ALTER EMPLOYEE_ID COL1 CONSTRAINT EMP_UNIQUE
  2> UNIQUE DEFERRABLE;
```

EMP_UNIQUE 制約は、EMPLOYEES テーブルから行を削除する場合には評価する必要がありません。

3.7.3 変更操作

変更するテーブルの列が制約の定義内で名前を指定されていない場合、Oracle Rdb は制約を評価する必要はありません。次に例を示します。

```
SQL> ALTER TABLE EMPLOYEES
  1> ALTER EMPLOYEE_ID CONSTRAINT EMP_ID_RANGE
  2> CHECK (EMPLOYEE_ID > '00000')
  3> NOT DEFERRABLE;
```

EMPLOYEES テーブルの EMPLOYEE_ID 以外の列を変更する場合は、Oracle Rdb で制約を評価する必要はありません。

制約内で名前を指定された列が COMPUTED BY フィールドの場合は、常にテーブルの任意の列を変更するたびに制約が評価されます。

変更が等価文の場合は、制約を評価する必要はありません。たとえば、次のクエリーを実行しても、Oracle Rdb は制約を評価しません。

```
SQL> UPDATE EMPLOYEES
  1> SET EMPLOYEE_ID = EMPLOYEE_ID;
```

3.7.4 データベース・キー（dbkey）による検索と消去

次に示す制約の定義について考察します。

```
SQL> ALTER TABLE EMPLOYEES
  1> ADD CONSTRAINT CONSTRAINT EMP_EXIST_SH
  2> CHECK (EMPLOYEE_ID = ANY (SELECT EMPLOYEE_ID
```

```
3> FROM SALARY_HISTORY))
4> DEFERRABLE;
```

Oracle Rdb では、検索する場合にデータベース・キーを使用してデータベース内の行の位置を確認します。この例では、EMPLOYEES テーブルに関して実行したクエリーによって、テーブルの行が一度に 1 行ずつ処理されます。Oracle Rdb は処理する各行の dbkey をすでに取得しているので、制約評価のメカニズムでは取得済みのデータベース・キーを使用して 1 行を検索し、制約を評価します。

3.8 ロック

データベース管理システムの基本的な機能は、データの整合性を保証できることです。多くのユーザーがデータベースに同時にアクセスすると、列の値が繰り返し変更される可能性があります。この列の変更がすべて終了すると、その列に最後に割り当てた値は最新の適切な値であることが期待されます。検索した列の最新の値を他のユーザーが破損しないように保護するメカニズムが存在しないと、データベース内のすべての値はほとんど信頼できません。Oracle Rdb ではロック・マネージャ[*1] を使用して共有リソースへのアクセスを同期化します。

ユーザー間のロックの競合はパフォーマンスの問題の原因の 1 つです。ロックの競合は、複数のトランザクションが同じリソースを同時にロックしようとした場合に発生します。Performance Monitor を使用すると、次の統計を表示できます。

- ロック（特定のロックのタイプ）
- ロックのサマリー統計（ロックの要求、ロックの昇格、ロックの降格、ロックの解放、非同期システム・トラップ（AST）のブロック、ストール時間、無効なロック・ブロック）
- 1 つの統計フィールドに関するロック統計（ロックの要求など）
- ロック統計（ファイルごと）

RMU Show Locks コマンドを使用すると、プロセスのロック・アクティビティに関する情報を表示できます。

OpenVMS VAX OpenVMS Alpha

OpenVMS の MONITOR LOCK コマンドも、データベース・ロックの問題の原因を判断するのに有効です。詳細は、OpenVMS のドキュメントを参照してください。◆

ロックに関する情報を収集するためのツールについては、3.8.1（3-44）項を参照してください。データベース・パフォーマンスを左右するロックに関する検討事項については、この章の後続の項を参照してください。ロックの問題を診断するチェックの手順については、8.4 項（8-43）を参照してください。

[*1] OpenVMS 上では、Oracle Rdb は OpenVMS ロック・マネージャを使用します。Compaq Tru64 UNIX 上では、Oracle Rdb は Oracle Rdb ロック・マネージャを使用します。

3.8.1 ロックに関する情報の収集

RMU Show コマンド、Performance Monitor の画面、System Dump Analyzer（SDA）ユーティリティ[*1] については、3.8.1.1（3-44）～ 3.8.1.6（3-59）項を参照してください。これらはすべてロックに関する情報を収集できます。3.2.1.3 項（3-9）と 3.2.1.5 項（3-16）で説明するストール・メッセージにも、ロックに関する情報が表示されます。ロックの分析に関するアドバイスは、8.4 項（8-43）を参照してください。

3.8.1.1 RMU Show Locks コマンド

RMU Show Locks コマンドを使用すると、特定のノード上に配置されたアクティブなデータベースのすべてについて、プロセスのロック・アクティビティとロックの競合に関する情報が表示されます。

クラスタ・システムでは、RMU Show Users コマンドによって現行ノードのみに関する詳細な情報が表示されますが、クラスタ規模の全般的な情報もいくつか取得できます。

OpenVMS システム上で RMU Show Locks コマンドを使用するには、WORLD 以上の OpenVMS プロセス権限が必要です。◆

Compaq Tru64 UNIX システム上で RMU Show Locks コマンドを使用するには、dbsmgr ユーザーまたはスーパー・ユーザーであることが必要です。◆

表 3-7（3-44）では、コマンド修飾子を示し、各修飾子について簡単に説明します。詳細は、この項の終わりに示す例に添付のテキストを参照してください。RMU Show Locks コマンドの詳細は、『Oracle RMU Reference Manual』を参照してください。

表 3-7 RMU Show Locks コマンド修飾子

修飾子	説明
Lock=(lock-id [, lock-id ...])	指定した 1 つまたは複数のロック ID に関する情報が表示されます。複数のロック ID を指定する場合は、カッコで囲み、カンマで区切る必要があります。ロック ID は 8 桁の 16 進数で、RMU Show Locks コマンドが発行されたノードに対してローカルでなければなりません。 Performance Monitor の「Stall Messages」画面を使用すると、プロセスの待機の原因になっているロック ID が表示されます。

[*1] OpenVMS システムのみ

表 3-7 RMU Show Locks コマンド修飾子（続き）

修飾子	説明
Mode = mode-list	表示するロックのモード（Blocking または Waiting）を指定します。 ■ Blocking モード・オプションを指定すると、自らのロックによって他のプロセスのロック要求をブロックしているプロセスが表示されます。出力の最初の行には、ロック要求が許可されるのを待機しているプロセスが表示されます。後続の行には、ロック要求が許可されるのを妨げているプロセスが表示されます。複数のプロセスが同じロック・リソースを待機している場合は、待機しているプロセスごとにプロセス固有の情報が表示されます。 ■ Waiting モード・オプションを指定すると、他のプロセスに許可されたロックの競合によって自らのロック要求が待機しているプロセスが表示されます。出力の最初の行には、リソース・ロックを許可されたプロセスが表示されます。後続の行には、同じリソースのロックを待機しているプロセスが表示されます。複数のプロセスが同じロック・リソースをブロックしている場合は、ブロックしているプロセスごとにプロセスに固有の情報が表示されます。 Blocking と Waiting の両方を指定する場合は、オプションをカンマで区切り、カッコで囲む必要があります。
Options = option-list	出力で表示する情報のタイプと詳細レベルを指定します。Options の 2 つの値は All と Full です。 ■ All オプションを指定すると、指定したプロセスが保持するロックに関係するすべてのプロセスに関するロック情報が表示されます。All オプションは必ず Process 修飾子と併用します。Mode 修飾子とは併用できません。 ■ Full オプションを使用すると、特殊データベース・プロセスに関するロック情報が表示されます。モニターなど、多くの特殊データベース・プロセスが、データベースに関する作業を行います。このようなデータベース・プロセスは頻繁にロックを要求し、設計上他のプロセスのロックと競合します。 Options に複数の値を指定する場合は、options-list をカンマで区切り、カッコで囲む必要があります。

表 3-7 RMU Show Locks コマンド修飾子（続き）

修飾子	説明
Output = file-name	出力を送信するファイルの名前を指定します。ファイル・タイプを指定しない場合、デフォルトの出力ファイル・タイプは .lis です。ファイル名を指定しないと、OpenVMS 上では SYS$OUTPUT に、Compaq Tru64 UNIX 上では標準出力（stdout）デバイスに出力が送信されます。
Process = (process-id [, process-id ...])	指定した 1 つまたは複数のプロセス ID に関するロック情報が表示されます。複数のプロセスを指定する場合は、ID をカッコで囲み、カンマで区切る必要があります。

RMU Show Locks コマンドに修飾子を指定しないと、ノード上のすべてのロックが表示されます。この表示は膨大になる恐れがあります。

[Ctrl] キーを押しながら [S] キーを押すと RMU Show Locks コマンドの出力を停止できます。[Ctrl] キーを押しながら [Q] キーを押すと出力を再開できます。RMU Show Locks コマンドを完全に終了する場合は、OpenVMS 上では [Ctrl] キーを押しながら [C] キーか [Y] キーを、Compaq Tru64 UNIX 上では [Ctrl] キーを押しながら [C] キーを押します。

表 3-8（3-46）に、コマンド修飾子をいくつか組み合せた結果を示します。この表はあらゆる組合せを示すわけではありません。この表に示すどの組合せを使用した場合も、同時に Options=Full を指定できますが、結果は変わりません。したがって、この表では Options=Full を省略します。

表 3-8 RMU Show Locks コマンド修飾子の組合せ

RMU Show Locks の修飾子の指定	表示されるロック情報
Process=process-id および Mode=Blocking	指定した 1 つまたは複数のプロセスをブロックしているすべてのプロセス
Process=process-id および Mode=Waiting	指定した 1 つまたは複数のプロセスを待機しているすべてのプロセス
Process=process-id および Options=All	指定した 1 つまたは複数のプロセスが保持するロックに関係するすべてのプロセス
Lock=lock-id および Mode=Blocking	指定した 1 つまたは複数のロックをブロックしているすべてのプロセス
Lock=lock-id および Mode=Waiting	指定した 1 つまたは複数のロックを待機しているすべてのプロセス

表 3-8 RMU Show Locks コマンド修飾子の組合せ（続き）

RMU Show Locks の修飾子の指定	表示されるロック情報
Mode=(Blocking,Waiting) および Process=(id-1,id-2)	プロセス id-1 をブロックしているすべてのプロセスと、プロセス id-2 を待機しているすべてのプロセス

プロセス ID またはロック ID のリストを指定する場合は、RMU Show Locks コマンドが発行されたノードのプロセスまたはロックに対してローカルであることを確認してください。

この項のこれ以降の部分では、いくつかの RMU Show Locks の修飾子の組合せについてサンプル・コマンドと出力を示します。各例には、2 つのトランザクション間の競合による出力を示します。最初のトランザクションのプロセス ID は 44A047C9 です。

```
SQL> ATTACH 'FILENAME mf_personnel';
SQL> SET TRANSACTION READ WRITE RESERVING EMPLOYEES FOR EXCLUSIVE WRITE;
```

2 番目のトランザクションのプロセス ID は 44A045D1 です。

```
SQL> ATTACH 'FILENAME mf_personnel';
SQL> SELECT * FROM EMPLOYEES WHERE LAST_NAME='Toliver';
```

2 番目のトランザクションは最初のトランザクションがコミットするかロール・バックするのを待機してからでないと実行できません。最初のトランザクションは EMPLOYEES テーブルに排他的にアクセスしており、2 番目のトランザクションはこのテーブルの読取りを要求しています。

生成する各レポートのヘディングは、コマンド行でどんな修飾子を使用したかを示します。レポートの書式と内容は、『Oracle RMU Reference Manual』を参照してください。Requested キューと Granted キューに表示できるロック・タイプの説明は、表 3-9（3-67）を参照してください。ただし、表 3-9（3-67）に示す SR ロックと SW ロックはそれぞれ RMU Show Locks のレポートに記載できる CR ロックおよび CW ロックと同等です。また、NL（NULL）ロックもレポートに記載されますが、表にはありません。NL ロックは、リソースへの関連を示すために、または今後のロック変換のプレースホルダとして使用されます。

例 3-13（3-48）に、RMU Show Locks コマンドに Process=44A047C9 修飾子を指定した場合に生成される出力の一部を示します。実際のレポートにはプロセス ID 44A047C9 で保持するすべてのロックが表示されるので、複数のページになります。レポートの本文には、ロックされているリソース、ID 情報、ロック・ステータス（Requested および Granted）が表示されます。

例 3-13 プロセスのロックの表示

```
================================================================================
SHOW LOCKS/PROCESS Information
================================================================================
.
.
.
--------------------------------------------------------------------------------
Resource: page 352
          ProcessID Process Name        Lock ID   System ID Requested Granted
          --------- ---------------     --------- --------- --------- -------
Owner:    44A047C9  USER1.........      7CC80BC8  00020025  PR        PR
--------------------------------------------------------------------------------
Resource: cluster membership
          ProcessID Process Name        Lock ID   System ID Requested Granted
          --------- ---------------     --------- --------- --------- -------
Owner:    44A047C9  USER1.........      16180C1A  00020025  PR        PR
.
.
.
--------------------------------------------------------------------------------
Resource: logical area 39
          ProcessID Process Name        Lock ID   System ID Requested Granted
          --------- ---------------     --------- --------- --------- -------
Owner:    44A047C9  USER1.........      45983EC0  00020025  EX        EX
.
.
.
--------------------------------------------------------------------------------
Resource: logical area 33
          ProcessID Process Name        Lock ID   System ID Requested Granted
          --------- ---------------     --------- --------- --------- -------
Owner:    44A047C9  USER1.........      0480973C  00020025  CR        NL
--------------------------------------------------------------------------------
Resource: logical area 53
          ProcessID Process Name        Lock ID   System ID Requested Granted
          --------- ---------------     --------- --------- --------- -------
Owner:    44A047C9  USER1.........      56009774  00020025  EX        EX
.
.
.
```

例 3-14（3-49）に、RMU Show Locks コマンドに修飾子 Process=44A047C9 と Mode=Waiting を指定した場合の出力を示します。このレポートは、指定したプロセス（44A047C9）が論理領域 39 に対して保持する排他的なロックが解放されるのを、プロセス ID 44A045D1 が待機していることを示しています。

このコマンドでは、待機しているプロセスが特定されます。自ら待機しているプロセスの ID を指定すると、次のメッセージが表示されます。"no locks on this node with the specified qualifiers."

例 3-14 ロックを待機しているプロセスの特定

```
================================================================================
SHOW LOCKS/PROCESS/WAITING Information
================================================================================
--------------------------------------------------------------------------------
Resource: logical area 39
          ProcessID Process Name        Lock ID   System ID Requested Granted
          --------- ---------------     --------- --------- --------- -------
Blocker:  44A047C9  USER1.........      45983EC0  00020025  EX        EX
Waiting:  44A045D1  _RTA11:........     3B5467DA  00020025  CR        NL
```

例 3-15（3-49）に、RMU Show Locks コマンドに修飾子 Process=44A045D1 と Mode=Blocking を指定した場合の出力を示します。このレポートは、プロセス ID 44A047C9 が論理領域 39 に対して排他的なロックを保持しており、指定したプロセス（44A045D1）をブロックしていることを示しています。

このコマンドでは、ブロックしているプロセスが特定されます。自らブロックしているプロセスの ID を指定すると、次のメッセージが表示されます。"no locks on this node with the specified qualifiers. "

例 3-15 ブロックしているプロセスの特定

```
================================================================================
SHOW LOCKS/BLOCKING Information
================================================================================
--------------------------------------------------------------------------------
Resource: logical area 39
          ProcessID Process Name        Lock ID   System ID Requested Granted
          --------- ---------------     --------- --------- --------- -------
Waiting:  44A045D1  _RTA11:........     3B5467DA  00020025  CR        NL
Blocker:  44A047C9  USER1.........      45983EC0  00020025  EX        EX
```

例 3-16（3-50）に、RMU Show Locks コマンドに修飾子 Lock=45983EC0 と Mode=Waiting を指定した場合の出力を示します。プロセス ID 44A047C9 によるロックは唯一なので、このレポートは例 3-14（3-49）の表示と同等です。プロセス ID 44A047C9 が複数のロックを保持する場合は、例 3-14（3-49）ではそのすべてが表示されますが、この例ではロック ID 45983EC0 に関する情報のみが表示されます。

例 3-16 待機しているプロセスのロック ID の表示

```
================================================================================
SHOW LOCKS/LOCK/WAITING Information
================================================================================
--------------------------------------------------------------------------------
Resource: logical area 39
          ProcessID Process Name        Lock ID   System ID Requested Granted
          --------- ---------------     --------- --------- --------- -------
Blocker:  44A047C9  USER1.........      45983EC0  00020025  EX        EX
Waiting:  44A045D1  _RTA11:........     3B5467DA  00020025  CR        NL
```

例 3-17（3-50）に、RMU Show Locks コマンドに修飾子 Process=44A047C9 と Options=All を指定した場合の出力の一部を示します。プロセス ID 44A047C9 でロックを保持するすべてのリソースと他のプロセスが同じリソース上で保持するすべてのロックが表示されるので、レポート全体では数ページに及びます。例 3-13（3-48）に示すレポートと比較してください。

例 3-17 プロセスがロックを保持するすべてのリソースの特定

```
================================================================================
SHOW LOCKS/PROCESS Information
================================================================================
.
.
.
--------------------------------------------------------------------------------
Resource: page 352
          ProcessID Process Name        Lock ID   System ID Requested Granted
          --------- ---------------     --------- --------- --------- -------
Owner:    44A047C9  USER1.........      7CC80BC8  00020025  PR        PR
Owner:    44A045D1  _RTA11:........     134C0979  00020025  PR        CR
--------------------------------------------------------------------------------
Resource: cluster membership
          ProcessID Process Name        Lock ID   System ID Requested Granted
          --------- ---------------     --------- --------- --------- -------
Owner:    44A047C9  USER1.........      16180C1A  00020025  PR        PR
Owner:    44A045D1  _RTA11:........     333C95ED  00020025  PR        PR
.
.
.
--------------------------------------------------------------------------------
Resource: logical area 39
          ProcessID Process Name        Lock ID   System ID Requested Granted
          --------- ---------------     --------- --------- --------- -------
Owner:    44A047C9  USER1.........      45983EC0  00020025  EX        EX
Waiting:  44A045D1  _RTA11:........     3B5467DA  00020025  CR        NL
.
```

```
.
.
-------------------------------------------------------------------------------
Resource: logical area 33
          ProcessID Process Name        Lock ID   System ID Requested Granted
          --------- ---------------     --------- --------- --------- -------
Owner:    44A047C9  USER1.........      0480973C  00020025  CR        NL
Owner:    44A045D1  _RTA11:.......      31900BAE  00020025  CR        CR
-------------------------------------------------------------------------------
Resource: logical area 53
          ProcessID Process Name        Lock ID   System ID Requested Granted
          --------- ---------------     --------- --------- --------- -------
Owner:    44A047C9  USER1.........      56009774  00020025  EX        EX
.
.
.
```

例 3-18（3-51）に、RMU Show Locks コマンドに修飾子 Mode=(Waiting,Blocker) と Process=(44A045D1,44A047C9) を指定した場合の出力を示します。このコマンドでは、例 3-15（3-49）と例 3-16（3-50）に示す出力を合成したレポートが表示されます。

例 3-18 待機しているプロセスとブロックしているプロセスの特定

```
===============================================================================
SHOW LOCKS/PROCESS/BLOCKING Information
===============================================================================
-------------------------------------------------------------------------------
Resource: logical area 39
          ProcessID Process Name        Lock ID   System ID Requested Granted
          --------- ---------------     --------- --------- --------- -------
Waiting:  44A045D1  _RTA11:.......      3B5467DA  00020025  CR        NL
Blocker:  44A047C9  USER1.........      45983EC0  00020025  EX        EX
===============================================================================
SHOW LOCKS/PROCESS/WAITING Information
===============================================================================
-------------------------------------------------------------------------------
Resource: logical area 39
          ProcessID Process Name        Lock ID   System ID Requested Granted
          --------- ---------------     --------- --------- --------- -------
Blocker:  44A047C9  USER1.........      45983EC0  00020025  EX        EX
Waiting:  44A045D1  _RTA11:.......      3B5467DA  00020025  CR        NL
```

RMU Show Locks による出力の表示での Granted モードと Requested モードの意味

RMU Show Locks コマンドを使用してロックを表示すると、指定したロックに対して Requested モードと Granted モードが表示されます。各モードの定義は次のとおりです。

- Requested
  これは、プロセスが要求したロックのモードです。有効なモードは NL、PR、PW、CR、CW、EX です。このモードは許可されるという保証がありません。一部のロックは、意図的に永久に競合モードの状態に保持されます（終了ロックなど）。
- Granted
  これは、プロセスが最後に許可されたロックのモードです。有効なモードは NL、PR、PW、CR、CW、EX です。ロックがまだ一度も許可されていない場合、ロック・モードは NL と表示されます。

RMU Show Locks コマンドの出力で Requested と Granted のロック・モードが異なる場合は、要求されたロックが Waiting キューか Conversion キューで現在ブロックされています。両方のモードが同じ場合は、ロックが許可されています。

ロック・マネージャは Requested ロック・モードを常に更新するわけではありません。すなわち、RMU Show Locks コマンドでは矛盾した情報が表示される可能性があります。

Requested ロック・モードは、次の場合にのみ更新されます。

1. ロック要求がリモート・リソースに対する場合
2. ロック要求が Nowait 要求の場合
3. ロックの競合によってロック要求が許可されなかった場合（アプリケーションによるロックの取消し、またはタイムアウトかデッドロックによる終了）
4. リソースに対する初めてのロック要求の場合

例 3-19（3-52）に、RMU Show Locks コマンドの出力の一部を示します。

例 3-19 RMU Show Locks コマンド

```
$ RMU/SHOW LOCKS
================================================================================
          SHOW LOCKS Information
================================================================================
.
.
.
--------------------------------------------------------------------------------
Resource Name: page 533
Granted Lock Count: 1,  Parent Lock ID: 01000B6C,   Lock Access Mode: Executive,
Resource Type: Global,  Lock Value Block: 03000000 00000000 00000000 00000002
          -Master Node Info-  --Lock Mode Information--     -Remote Node Info-
ProcessID Lock ID   SystemID  Requested Granted   Queue     Lock ID   SystemID
2040021E  0400136A  00010002  EX        CR        GRANT     0400136A  00010002
--------------------------------------------------------------------------------
.
.
.
```

通常、例 3-19（3-52）に示すロック・モードの組合せがどのように発生したのかを説明するのは困難です。CR（同時読取り）モードが Grant キューにあること（Conversion キューでなく）に注意してください。

このシステムにはノードが 1 つしかないことを認識するには、動作環境に関する知識が必要です。この出力を生成するために 2 つのロック要求が発生しましたが、実際とは逆の順序で発生したように見えます。

最初のロック要求は EX（排他）モードです。これはただちに許可されました。したがって、Requested モードと Granted モードは 4 番目の状況に従って更新されています。ここで、ロックは EX モードから CR モードに降格され、これもただちに許可されました。しかし、前述の 4 つの条件はいずれも真でないので「Requested」フィールドは更新されません。Requested モードが CR ロックの要求を反映して更新されることはありません。

3.8.1.2 総合的なロック情報の表示

Performance Monitor では、「Stall Messages」、「Active User Stall Messages」、「DBR Activity」の各画面に総合的なロック情報が表示されます。

[L] を入力すると、アクティブな画面に現在表示されている任意のロック ID のメニューが表示されます。

例として、次に示す Performance Monitor の「Active User Stall Messages」画面について考えます。

```
Node: TRIXIE         Oracle Rdb V7.0-00 Performance Monitor 16-MAY-1996 14:48:03
Rate: 1.00 Second          Active User Stall Messages       Elapsed: 03:06:07.20
Page: 1 of 1     KODD$:[R_ANDERSON.WORK.AIJ]MF_PERSONNEL.RDB;1      Mode: Online
-------------------------------------------------------------------------------
Process.ID Since......   Stall.reason.............................. Lock.ID.
2B600554:1 15:38:27.04 - waiting for record 55:321:2 (EX)          1500624B
2B600556:1 15:40:40.66 - waiting for record 55:321:2 (EX)          1200A2E4
2B600555:1               writing pages back to database
-------------------------------------------------------------------------------
Config Exit Help LockID Menu >next_page <prev_page Set_rate Write Zoom !
```

[L] を入力すると、有効なロック ID のメニューが表示されます。次に例を示します。

```
Node: TRIXIE         Oracle Rdb V7.0-00 Performance Monitor 16-MAY-1996 14:48:11
Rate: 1.00 Second          Active User Stall Messages       Elapsed: 03:06:07.20
Page: 1 of 1     KODD$:[R_ANDERSON.WORK.AIJ]MF_PERSONNEL.RDB;1      Mode: Online
-------------------------------------------------------------------------------
Process.ID Since......   Stall.reason.............................. Lock.ID.
2B600554:1 15:38:27.04 - waiting for record 55:321:2 (EX)          A. 1500624B
2B600556:1 15:40:40.66 - waiting for record 55:321:2 (EX)          B. 1200A2E4
2B600555:1               writing pages back to database
-------------------------------------------------------------------------------
```

```
Config Exit Help LockID Menu >next_page <prev_page Set_rate Write Zoom !
```

ただし、ロック ID の前に標準のメニュー選択文字が表示されています。任意の文字を選択するか [↑] キーと [↓] キーを使用してハイライトしたカーソルを移動してから [Enter] キーを押し、任意のロック ID を選択します。

ロック ID が表示されていない場合は、[L] を入力しても何も起こりません。

任意のロック ID を選択すると、「Lock Information」表示が表示されます。次に例を示します。

```
Node: TRIXIE         Oracle Rdb V7.0-00 Performance Monitor 16-MAY-1996 14:48:15
Rate: 1.00 Second          Active User Stall Messages       Elapsed: 03:06:07.20
Page: 1 of 1     KODD$:[R_ANDERSON.WORK.AIJ]MF_PERSONNEL.RDB;1      Mode: Online
--------------------------------------------------------------------------------
Process.ID Since......   Stall.reason.............................. Lock.ID.
2B600554:1 15:38:27.04 - waiting for record 55:321:2 (EX)         >1500624B<
2B600556:1 15:40:40.66 - waiting for record 55:321:2 (EX)          1200A2E4
2B600555:1               writing pages back to database
+-- Lock Information: 1500624B -----------------------------+
|                                                           |
|              Resource: record 321:2                       |
| State... ProcessID Process.Name... Lock.ID.  Rq Gr Queue  |
|                                                           |
| Blocker: 2B600555  RICK3.......... 0F005AB6  EX EX Grant  |
| Waiting: 2B600556  RICK4.......... 1200A2E4  EX NL Cnvrt  |
| Waiting: 2B600554  RICK2.......... 1500624B  EX NL Cnvrt  |
|                                                           |
+-----------------------------------------------------------+
--------------------------------------------------------------------------------
Config Exit Help LockID Menu >next_page <prev_page Set_rate Write Zoom !
```

「Lock Information」表示は動的ではありません。任意のロック ID を選択するとロック情報が獲得されますが、この情報は動的に変化するわけではありません。「Lock Information」表示は、ロック ID を選択した時点でのスナップショットです。

「Lock Information」表示は表示専用です。Write オプションを使用して保存することはできません。

「Lock Information」表示には、次の情報が表示されます。

情報	ステータス
State	個々のロックがリソースを所有しているか、ブロックしているか、または待機しているか。
ProcessID	個々のロックを所有するプロセスの ID。
Process.Name	個々のロックを所有するプロセスの名前。

情報	ステータス
Lock.ID	表示されたリソース上のロックの ID。
Rq	個々のロックが要求されたモード（NL、CR、CW、PR、PW、EX）。
Gr	個々のロックが許可されたモード（NL、CR、CW、PR、PW、EX）。許可されたロックは要求されたロック・モードと異なる場合があります（Conversion キューにある場合または強力なロックから緩やかなロックに降格された場合）。
Queue	個々のロックが存在するキューには次の 3 つのキューがあります。 ■ Grant - Grant キュー ■ Cnvrt - Conversion キュー ■ Wait - Waiting キュー

ロックの表示順序は不定ですが、待機しているロックが許可される順序は特定できることに注意してください。たとえば、前の例では 2 つのプロセスが同一のロックを待機しています。ロック 1500624B に関する情報では、2 つのプロセスが同じリソースを待機していることだけが示されます。ロック 1200A2E4 に関する表示では、次の情報が表示されます。

```
Node: TRIXIE         Oracle Rdb V7.0-00 Performance Monitor 16-MAY-1996 14:49:01
Rate: 1.00 Second          Active User Stall Messages       Elapsed: 03:06:07.20
Page: 1 of 1     KODD$:[R_ANDERSON.WORK.AIJ]MF_PERSONNEL.RDB;1      Mode: Online
--------------------------------------------------------------------------------
Process.ID Since......   Stall.reason.............................. Lock.ID.
2B600554:1 15:38:27.04 - waiting for record 55:321:2 (EX)           1500624B
2B600556:1 15:40:40.66 - waiting for record 55:321:2 (EX)          >1200A2E4<
2B600555:1               writing pages back to database
+-- Lock Information: 1500624B ----------------------------+
|                                                          |
|              Resource: record 321:2                      |
| State... ProcessID Process.Name... Lock.ID.  Rq Gr Queue |
|                                                          |
| Blocker: 2B600555  RICK3.......... 0F005AB6  EX EX Grant |
| Blocker: 2B600554  RICK2.......... 1500624B  EX NL Cnvrt |
| Waiting: 2B600556  RICK4.......... 1200A2E4  EX NL Cnvrt |
|                                                          |
+----------------------------------------------------------+
--------------------------------------------------------------------------------
Config Exit Help LockID Menu >next_page <prev_page Set_rate Write Zoom !
```

この表示は、ロック 1200A2E4 は Conversion キューにあるが他方のロックをブロックしており、他方のロックの前に許可されることを示します。

「Lock Information」表示には、端末画面には表示できない情報も表示されます。したがって、「Lock Information」表示は一度に 1 行または一度に 1 画面ずつスクロールできます。

Lock Information 画面では、[↑] キーと [↓] キーを使用して一度に 1 行ずつ移動できます。また、[←] キーか [Page Up] キー、または [→] キーか [Page Down] キーを使用すると、一度に 1 ページずつ移動できます。インジケータ "<--MORE" と "MORE-->" がロック情報の表示自体の上に動的に表示される場合は、上記のキーがアクティブになっています。画面下部の水平メニューでも、上記のキーがアクティブかどうかがわかります。ナビゲーショナル・キーがアクティブでない場合は、上記のキーを押しても何も起こりません。

「Lock Information」表示のサイズは、端末の行数の構成に基づいています。たとえば、48 行の端末では 24 行の端末より「Lock Information」表示が大きくなります。

ロック情報を表示している間もデータベースは動作を継続するので、選択したロックの情報が表示される前にそのロックが解放されることもあります。この場合は、端末画面にメッセージ "Lock has been released since it was displayed" が表示されます。また、「Lock Information」画面から戻るとロック ID メニューは取り消されます。

ロック情報を検索するための権限が十分でない場合は、端末の画面にメッセージ "Insufficient privilege to display lock information" が表示されます。また、「Lock Information」画面から戻るとロック ID メニューは取り消されます。

ロック情報には動的な性質があるので、ロック情報は出力ファイルには書き込まれません。したがって、入力ファイルから再実行することもできません。

「Lock Information」表示が表示されている間は、端末のブロードキャスト・メッセージは一時停止します。「Lock Information」表示を終了すると、ブロードキャスト・メッセージが再開されます。

「Active User Stall Messages」画面には、過去にストールされたが現在はストールされていないロックも表示されるので注意してください。表示されているロックが現在はストールされていない場合も、ストールされたロックに関する情報を表示できます。ただし、過去にストールされたロックの多くは解放されています。

「Lock Information」表示には、個々のロックを保持するプロセスの詳細は表示されません。プロセスの情報は、プロセスが現行ノード上にある場合は「Process Accounting」画面に表示されます。

現在、分割のないメニューにはオプション数 36 の制限があるので（A ～ Z と 0 ～ 9 による）、表示された画面の最初の 36 のロック ID のみを選択できます。将来はこの制限を解除する見込みです。この制限を回避するには、端末画面のサイズを 42 行未満に設定するか、別の「Stall Messages」画面で現在の画面とは異なる順序でロック ID を表示します。

3.8.1.3 Performance Monitor によるロック統計の収集

Performance Monitor では、次の画面を使用してロックに関する情報を収集できます。

- 「Summary Locking Statistics」画面。Oracle Rdb で実行するロック・アクティビティのサマリーが表示されます。
- 「Locking for One Lock Type」画面。特定のロック・タイプに関するロックの統計が表示されます。この画面は、特定のロック・タイプに関するすべての情報を表示する場合に使用します。このロック画面が「Lock Statistics for One Statistics Field」画面と唯一異なる点は、指定した唯一のロック・タイプに関する統計が表示されることです。したがって、ロック・アクティビティをより詳細に分析するのに便利です。
- 「Locking for One Statistics Field」画面。すべてのロック・タイプに関して特定の統計分野のロック統計が表示されます。この画面は、様々なロック・タイプに関する特定の情報を表示する場合に使用します。
- 「Lock Deadlock History」画面。デッドロック・イベントの原因となるオブジェクトの特定に使用します。
詳細は、3.8.1.4 項（3-57）を参照してください。
- 「Lock Timeout History」画面。タイムアウト・イベントの原因となるオブジェクトの特定に使用します。
詳細は、3.8.1.5 項（3-58）を参照してください。
- 「Lock Statistics by file」画面。記憶領域とスナップショット・ファイルに固有のページのロックに関する情報が表示されます。ロック・アクティビティの多い記憶領域を検出し、記憶領域のパーティションの妥当性を分析する場合には、この情報が特に重要です。

3.8.1.4 Performance Monitor の「Lock Deadlock History」画面

「Lock Deadlock History」画面は、デッドロック・イベントの原因となるオブジェクトの特定に使用します。「Lock Deadlock History」画面は、「Process Information」サブメニューから表示します。

「Stall Messages」画面にはストールの情報が表示されますが、ロックがデッドロックするとストールが終了し、「Stall Messages」画面にはこの情報が表示されなくなります。

「Lock Deadlock History」画面には、現行ノード上のアクティブなプロセスごとに、プロセス ID、プロセスが最後にデッドロックした時刻、最後のデッドロックの原因、データベースに接続してからプロセスに発生したデッドロックの回数が表示されます。

次の「Lock Deadlock History」画面は、レコードのデッドロックを示しています。ロックのデッドロックの原因ではプロセスがレコードを待機していたことを示しており、そのレコードの dbkey が表示されています。

```
Node: TRIXIE         Oracle Rdb V7.0-00 Performance Monitor 16-MAY-1996 14:48:12
Page: 0.10 Seconds           Lock Deadlock History          Elapsed: 00:07:21.10
Page: 1 of 3           SQL_DISK1:[RICK.V70]MF_PERSONNEL.RDB;1       Mode: Online
--------------------------------------------------------------------------------
Process.ID Occurred...   Lock.timeout.reason..................... #Timeout
65505487:1 08:55:21.16   Waiting for record 1:2:1 (CR)                    1
--------------------------------------------------------------------------------
Config Exit Help Menu >next_page <prev_page Set_rate Write Zoom !
```

DBA は、「Lock Deadlock History」画面で最新のデッドロックについて調べれば、データベースのホット・スポットとアプリケーションのボトルネックの可能性を容易に識別できます。また、画面全体に表示されたデッドロックの情報に注目すれば、頻繁に発生する問題とそれ以外の問題を区別し、問題の一般的な原因の相互関係を調べることができます。

次の「Lock Deadlock History」画面は、ページのデッドロックを示しています。ページのデッドロックは問題であるとは限りませんが、パフォーマンスの問題の可能性を示す場合があります。プロセス 7660441F には 36 のデッドロックがありましたが、必ずしも表示されたページに関するものではありません（ロックのデッドロックの原因は最新のデッドロックの原因だけを示します）。プロセスがデータベースに接続された時刻に対して 36 のデッドロックが多い場合、DBA はこのプロセスを詳細に調べる必要があります。特定のプロセスに関するデッドロックの数が多い場合は、設計上の大きな問題を示している可能性があります。

```
Node: TRIXIE         Oracle Rdb V7.0-00 Performance Monitor 16-MAY-1996 14:51:33
Page: 0.10 Seconds           Lock Deadlock History          Elapsed: 00:07:23.12
Page: 1 of 3           SQL_DISK1:[RICK.V70]MF_PERSONNEL.RDB;1       Mode: Online
--------------------------------------------------------------------------------
Process.ID Occurred...   Lock.deadlock.reason.................... #Deadlock
76601621:1                                                                 0
7660441F:1 14:09:01.89 - waiting for page 1:258 (PR)                      36
--------------------------------------------------------------------------------
Config Exit Help Menu >next_page <prev_page Set_rate Write Zoom !
```

一般に、レコードのデッドロックは好ましくないので、原因を突き止めて対処する必要があります。ページのデッドロックは問題であるとは限りませんが、分析は必要です。

3.8.1.5 Performance Monitor の「Lock Timeout History」画面

「Lock Timeout History」画面は、タイムアウト・イベントの原因となるオブジェクトの特定に使用します。「Lock Timeout History」画面には、「Process Information」サブメニューからアクセスします。

「Stall Messages」画面にはストールの情報が表示されますが、プロセスがロックを待機する間にタイムアウトになるとストールが終了し、「Stall Messages」画面にはこの情報が表示されなくなります。

「Lock Timeout History」画面には、現行ノード上のアクティブなプロセスごとに、プロセス ID、プロセスがロックの待機によって最後にタイムアウトした時刻、プロセスに最後に発生

したタイムアウトの原因、データベースに接続してからプロセスに発生したタイムアウトの回数が表示されます。

```
Node: TRIXIE         Oracle Rdb V7.0-00 Performance Monitor 16-MAY-1996 14:54:13
Rate: 3.00 seconds            Lock Timeout History          Elapsed: 00:00:31.75
Page: 1 of 3         SQL_DISK1:[RICK.V70]MF_PERSONNEL.RDB;1         Mode: Online
------------------------------------------------------------------------------
Process.ID Occurred...   Lock.timeout.reason..................... #Timeout
65505487:1 08:55:21.16   Waiting for record 1:2:1 (CR)                    1
------------------------------------------------------------------------------
Config Exit Help Menu >next_page <prev_page Set_rate Write Zoom !
```

3.8.1.4 項（3-57）の最初の「Lock Deadlock History」画面の例とこの「Lock Timeout History」画面を調べると、dbkey 1:2:1 のレコードがホット・スポットであり、競合の原因になっていることがわかります。

3.8.1.6 System Dump Analyzer

OpenVMS VAX OpenVMS Alpha

ロックについて詳しく調べる場合は、System Dump Analyzer（SDA）ユーティリティを使用します。詳細は OpenVMS のドキュメントを参照してください。

SDA を使用する場合は、次の手順に従います。

1. SDA を使用するには、OpenVMS の CMKRNL 権限が必要です。CMKRNL 権限は強力なので、必要のないユーザーには付与しないでください。
2. 「Stall Messages」画面か「Active User Stall Messages」画面を使用すると、ストールしたプロセスのプロセス ID（PID）を抽出できます。
3. 次のように SDA を起動します。

   ```
   $ ANALYZE/SYSTEM
   ```

4. ストールしたプロセスの PID の値を表示します。

   ```
   SDA> SHOW SUMMARY
   ```

5. ストールしたプロセスを PID で選択します。

   ```
   SDA> SET PROCESS/INDEX=index_value
   ```

6. プロセスの選択が正しいことを確認します。

   ```
   SDA> SHOW PROCESS
   ```

7. 後で編集できる出力ファイルを作成します。

   ```
   SDA> SET OUTPUT file_name.ext
   ```

8. プロセスの情報を表示します。

   ```
   SDA> SHOW PROCESS/INDEX=index_value/LOCKS
   ```

9. 後続のコマンドの出力を SYS$OUTPUT にリダイレクトし、プロセスの情報を記載したファイルを閉じます。

```
SDA> SET OUTPUT SYS$OUTPUT
```

10. サブプロセスを起動し、ロック情報で作成したファイルを編集できるようにします。

```
SDA> SPAWN
```

11. テキスト文字列 "Waiting" を検索します（ファイルの末尾から開始）。

```
$ EDIT file_name.ext
```

12. この特定のロックのロック ID を書き留め、サブプロセスをログアウトします。

```
$ LO   (get back to SDA)
```

13. 特定のロックに関する情報を表示します。

```
SDA> SHOW RESOURCE/LOCK=lock_id
```

これで、許可されているロック、待機しているロック、ロックしているリソースが表示されます。LOCK QUEUE の情報をすべて書き留めます。

14. PID の値を決定します。

```
SDA> SHOW LOCK lock_id
```

15. 有効なプロセス ID の値を指定します。

```
SDA> SHOW PROCESS/INDEX=pid_from_previous_step
```

16. 終了します。

```
SDA> EXIT
```

17. ストールの原因になったプロセス、または原因の一端となったプロセスを表示します。

```
$ SHOW PROCESS/ID=Process_ID_value
```

ストールしたプロセスごとに、以上の処理を繰り返す必要があります。

バッチ・ジョブによって問題が発生している場合は、そのソフトウェアのロジックを変更しなければならないことがあります。ほとんどの場合、トランザクションは NOWAIT で開始されており、ソフトウェア・トラップ・エラーが発生するので再試行がスケジュールされます。Oracle Rdb のデフォルトでは、トランザクションは WAIT で開始されます。このことは、ストールの状態に少なからず関連します。

ソート・インデックスの使用とそれに伴うインデックス・ノードのサイズも、この問題に関連する場合があります。かわりにハッシュ・インデックスを使用するのは有効な方法です。それ以外の場合は、インデックス・ノードのサイズを縮小します。

ロックを保持するプロセスを一時停止することもできます。このためには、STOP コマンドや EXIT コマンドでなく [Ctrl] キーと [Y] キーを同時に押すか、バッチ・ジョブを一時停止します。◆

3.8.2 ロックに関する検討事項

1 人のユーザーのクエリーによって読取り / 書込みトランザクションでデータベースから 1 行をフェッチする場合は、一連の読取り保護（PR）または書込み保護（PW）のロックによって他のユーザーがその行の変更を阻止されます。Oracle Rdb では、その行だけでなくデータベース・テーブル、ページ、インデックス・ノードにもこのようなロックを適用します。後続の操作でこの行を検索しようとしても、この行に関する現行ユーザーの操作と競合することはありません。しかし、ユーザーのクエリーで読取り専用のデータベースから 1 行をフェッチする場合、このクエリーでは論理領域上の共有読取り（SR）ロックを使用するので、他のユーザーは論理領域上でトランザクションがアクティブなことを認識します。行レベルのロックは実行されませんが、読取り専用トランザクションではページ・ロックを使用してデータベース・ページのインスタンスが各ユーザーのキャッシュで有効なことを保証します。Oracle Rdb のロック・アクティビティの詳細は、『Oracle Rdb7 Guide to Database Design and Definition』を参照してください。

Oracle Rdb は、トランザクションによるデータベース・リソースへのアクセスを制御します。トランザクションを開始する場合は、実行するデータベース・アクティビティの種類を指定します。使用するテーブルへのアクセスを制限したり、他のユーザーが同じデータを共有するのを許可したりできます。トランザクションを開始すると、トランザクションの適用範囲で実行するすべての操作は整合性が保証されます。すなわち、ユーザーが検索または更新する行はトランザクションが持続する間は安定です。トランザクションが終了すると、更新された行は他のデータベース・ユーザーが使用できるようになります。しかし、トランザクション進行中にユーザーが介入して行を変更することはできません。

一連のデータベース操作を正常に完了できるトランザクション・モードを指定すると、ユーザーのトランザクションが競合する同時のデータベース・アクティビティのタイプに影響することがあります。アプリケーションでマルチユーザーによる同時読取りアクセスを要求する場合は、スナップショット・ファイルを有効にし、読取り専用トランザクションを使用すると、バッチ更新トランザクションとロック指定の句（RESERVING...FOR EXCLUSIVE...）を使用したトランザクション以外のトランザクション・タイプとの競合を回避できます。他の多くのケースで更新アクセスが必要な場合は、読取り / 書込みトランザクションが使用されます。3.8.3 項（3-62）に、SQL の SET TRANSACTION 文に記述できるデータベースのアクセス・モードについて説明します。SQL の SET TRANSACTION 文の詳細と様々なトランザクション・タイプ間のロック互換性の詳細は、『Oracle Rdb7 SQL Reference Manual』を参照してください。

3.8.3 予約オプション

最適なトランザクションの設計では、ロックが最低限に保持されます。SQL の SET TRANSACTION 文で予約オプションを使用すると、タスクに必要なデータベースの並列性とセキュリティのレベルを実現できます。予約オプションは、共有モードとロック・タイプ（読取りまたは書込み）を組み合せたものです。共有モード・オプションは次のとおりです。

- Shared（共有）
- Protected（保護）
- Exclusive（排他）

データベースへのマルチユーザー・アクセスは使用するアプリケーションとデータによって増減するので、データベースを同時に使用するユーザーの数を予想し、この数を指針としてアクセス・モードを決定する必要があります。

シングルユーザー・アクセスでデータベースの大規模な更新を行うようにスケジュールする場合は、SQL の SET TRANSACTION 文で排他的書込みモードを指定することで、ロックするリソースを最小限に抑え、パフォーマンスを向上できます。排他的書込みモードでは、スナップショット・ファイルへの書込みや行レベルのロック情報の保持によるオーバーヘッドはありません。したがって、トランザクションで排他的書込みモードを使用すると、同じトランザクションで他の共有モードとロック・タイプを使用した場合よりも短時間で終了し、使用するメモリーも小さくなります。

マルチユーザーによる同時アクセスが必要なアプリケーションの場合、アクセス・モードの選択はクエリーのタイプによって変わります。読取りのアクセスだけが必要なクエリーの場合は、読取り専用トランザクションを使用します。データベースを更新するクエリーの場合は、可能であれば共有書込みモードを使用します。共有書込みモードでは並列性が向上します。書込み保護モードでは並列性が低下し、指定したテーブルに対してシングルユーザーによる更新のみが許可されます。他のユーザーが保護テーブルに書き込もうとすると異常終了したり待機したりするので、ロール・バックが必要になります。読取り専用トランザクションを要求するユーザーは、このテーブルへのアクセスを許可されます。

多くの同時ユーザーによる更新が必要なタスクの場合は、SQL の SET TRANSACTION 文で共有モードを使用します。他のユーザーはテーブルを更新できます。共有書込みモードではテーブルを同時に更新できるユーザーの数は増えますが、すべてのユーザーによるロックの競合も増大します。デッドロックが発生し、作業の再実行を余儀なくされる場合があります。共有書込みモードを使用するとトランザクション保護は強化されますが、ロックの競合が解決するまでの待ち時間が長くなるので、トランザクション全体の所要時間も若干長くなります。

例 3-20（3-63）、例 3-21（3-64）、例 3-24（3-66）、例 3-25（3-66）、例 3-26（3-67）に、予約オプションと他のトランザクションとの互換性のレベルを示します。

例 3-20 共有読取り予約オプションによるトランザクションの開始

```
SQL> ATTACH 'FILENAME mf_personnel';
SQL> SET TRANSACTION READ WRITE RESERVING
  1> EMPLOYEES FOR SHARED READ;
```

例 3-20（3-63）に示す共有読取り要求では、すべての検索トランザクションから特定のデータベース・リソースへのアクセスが許可されます。このモードで予約されたテーブルを更新する他のトランザクションも、同じリソースへのアクセスを許可されます。共有読取りと共有書込みは、書込みが実際にリソースを更新するまでは同じリソースにアクセスできます。

物理領域が、1 つのトランザクションでは共有読取りモードでレディとなり、次のトランザクションでは共有書込みモードでレディの場合は、2 番目のトランザクションがこの領域でレディとなるときには、この領域のバッファはフラッシュされません。

これ以外の物理領域のモード変換では、2 番目のトランザクションが領域を準備するときに、Oracle Rdb が必ずその領域のバッファをフラッシュします。このモード変換がデータ領域を対象とする場合、Oracle Rdb はまずそのスナップショット領域のバッファをフラッシュし、次にデータ領域のバッファをフラッシュし、最後に必要な場合はロックを降格または昇格します。

テーブルの共有書込みまたは共有読取りを予約した場合の動作は、ロックの観点から見ると次の例外を除いて予約句を指定しない場合の動作と同じです。

- 共有読取りトランザクションの予約。常に CR（同時読取り）モードでロックされ、トランザクションによるこのテーブルへの書込みを阻止します。
- 共有読取り予約と共有書込み予約のトランザクションは、保護予約または排他的共有モード予約のトランザクションによってデッドロックすることはありません。

テーブルが予約されると、テーブルのすべての論理領域はトランザクションを開始する前の準備を省略されます。かわりに、トランザクション開始時に 1 つの論理領域をロックして保護トランザクションや排他的トランザクションが実行されないようにする必要があります。後続の論理領域にアクセスするとその領域がロックされます。これは予約句を指定しない場合と同様です。したがって、最悪の場合、共有読取り予約または共有書込み予約のトランザクションでは、トランザクションの間にアクセスするテーブルごとにもう 1 つのロック操作が必要になります。

更新トランザクションは、同じテーブルへのアクセスを共有します。行を更新するときのロックの競合を最小限に抑えるには、いくつかの方法があります。行を更新するがそのインデックスを更新しない場合は、DISCONNECT 文を発行するまで dbkey が有効であるように指定します。データベースの最初の宣言または起動時に、SQL の DBKEY SCOPE IS ATTACH 句を使用してください。このユーザーがデータベースの接続を解除するまで（通常は DISCONNECT 文による）、各行の dbkey が変更されないことが保証されます（行が削除された場合も）。

ロックの競合を最小限に抑えるために、変更する行とその dbkey を検出して読取り専用トランザクションを開始します。COMMIT 文を発行します。ここで、例 3-21（3-64）に示すよ

うに読取り専用トランザクションを開始し、変更する行をその dbkey で参照し、必要な変更を行います。これは、同時環境で多くの更新が必要な場合には適切な方法です。

例 3-21 共有書込みモードによるトランザクションの開始

```
SQL> ATTACH 'FILENAME mf_personnel';
SQL> SET TRANSACTION READ WRITE RESERVING
  1> EMPLOYEES FOR SHARED WRITE;
```

I/O 操作に費用がかかるので、更新件数が少ない場合やテーブルのインデックスが変わる場合はこの方法は実用的ではありません。したがって、行の更新でインデックスの要素になっている列を変更しない場合は、その dbkey で行を検索するとロックの競合が軽減されます。

例として、mf_personnel サンプル・データベースの EMPLOYEES テーブルについて考察します。ある従業員の行を検索し、その住所を変更する場合を想定します。プログラムで行の dbkey を保持すれば、例 3-22（3-64）の構文を使用できます。

例 3-22 dbkey による行の更新と検索

```
SQL> ATTACH 'FILENAME MF_PERSONNEL.RDB';
SQL> --
SQL> -- Declare host variables
SQL> --
SQL> DECLARE :hv_row INTEGER;               -- Row counter
SQL> DECLARE :hv_employee_id ID_DOM;        -- EMPLOYEE_ID field
SQL> DECLARE :hv_employee_id_ind SMALLINT; -- Null indicator variable
SQL> --
SQL> DECLARE :hv_dbkey CHAR(8);             -- DBKEY storage
SQL> DECLARE :hv_dbkey_ind SMALLINT;        -- Null indicator variable
SQL> --
SQL> DECLARE :hv_last_name LAST_NAME_DOM;
SQL> DECLARE :hv_new_address_data_1 ADDRESS_DATA_1_DOM;
SQL> --
SQL> -- Set host variables
SQL> --
SQL> SET TRANSACTION READ WRITE;
SQL> BEGIN
  1> --
  2> -- Set the search value for SELECT
  3> --
  4>  SET :hv_last_name = 'Ames';
  5> --
  6> -- Set the NEW_ADDRESS_DATA_1 value
  7> --
  8> SET :hv_new_address_data_1 = '100 Broadway Ave.';
  9> END;
SQL> COMMIT;
SQL> --
```

```
SQL> SET TRANSACTION READ ONLY;
SQL> BEGIN
  1> SELECT E.EMPLOYEE_ID, E.DBKEY
  2>   INTO :hv_employee_id INDICATOR :hv_employee_id_ind,
  3>        :hv_dbkey INDICATOR :hv_dbkey_ind
  4>   FROM EMPLOYEES E
  5> WHERE E.LAST_NAME = :hv_last_name
  6> LIMIT TO 1 ROW;
  7> --
  8> GET DIAGNOSTICS :hv_row = ROW_COUNT;
  9> END;
SQL> COMMIT;
SQL> --
SQL> SET TRANSACTION READ WRITE RESERVING EMPLOYEES FOR SHARED WRITE;
SQL> BEGIN
  1> IF (:hv_row = 1) THEN
  2>    BEGIN
  3>    UPDATE EMPLOYEES E
  4>      SET E.ADDRESS_DATA_1 = :hv_new_address_data_1
  5>    WHERE E.DBKEY = :hv_dbkey;
  6>    END;
  7> END IF;
  8> END;
SQL> COMMIT;
SQL> --
SQL> -- Display result of change
SQL> --
SQL> SET TRANSACTION READ ONLY;
SQL> SELECT E.*
  1> FROM EMPLOYEES E
  2> WHERE E.DBKEY = :hv_dbkey;
 EMPLOYEE_ID   LAST_NAME        FIRST_NAME   MIDDLE_INITIAL
   ADDRESS_DATA_1              ADDRESS_DATA_2         CITY
      STATE   POSTAL_CODE   SEX    BIRTHDAY      STATUS_CODE
 00416         Ames             Louie        A
   100 Broadway Ave.                                  Alton
      NH      03809         M      13-Apr-1941   1
1 row selected
SQL>
```

例 3-22（3-64）では、データ行だけがロックされます。

EMPLOYEE_ID 列（例 3-23（3-66）を参照）の値に基づいてプログラムで行をフェッチする場合は、その行に到達するまでに横断する EMPLOYEE_ID のインデックスに関するインデックス・ノードをロックする必要があります。データ行もロックします。EMPLOYEE_ID にインデックスがない場合は、行を検出するのにテーブル全体がロックされます。

例 3-23 列の値による行の更新と検索

```
SQL> SELECT E.EMPLOYEE_ID
  1>  FROM EMPLOYEES E
  2>  WHERE E.EMPLOYEE_ID=<some ID number>;
SQL> UPDATE EMPLOYEES
  1>  SET ADDRESS_DATA_1 = <new address>
  2>  WHERE EMPLOYEE_ID=<some ID number>;
```

したがって、例 3-22（3-64）で dbkey によるフェッチを行うと、ロックが競合する確率が低くなります。インデックスがロックされないので、ロックの競合が少なくなります。

dbkey を使用して行を更新しても、更新する列がインデックスに使用されている場合は、インデックスを更新する必要があります。したがって、インデックス・ノードがロックされることになります。この場合は、dbkey によるフェッチでもロックの競合が減少しません。dbkey で検索するか列の値で検索するかにかかわらず、同じロックが保持されます。

ロックの競合を軽減するもう 1 つの方法は、事前にコンパイルした SQL またはモジュール言語プログラムを使用して、同じデータベースを 2 つの別々のエイリアスで 2 度宣言することです。DECLARE ALIAS 文で DBKEY SCOPE IS ATTACH 句を指定し、一部の行が削除されても dbkey が常に同じ行を指していることを保証する必要があります。したがって、行を削除したユーザーが DISCONNECT 文を発行するまで、Oracle Rdb は削除した行の dbkey を再使用することはできません。この方法は、データベースを読取り / 書込みと読取り専用トランザクションで一度ずつ取り込む場合、個々の更新トランザクションを実行するのに便利です。dbkey を検出したり、またデータベースの読取り専用コピーの検索を実行してから、読取り / 書込みコピーを更新します。この方法では、システム・リソースを大量に使用しますが、ロックの競合は著しく減少します。詳細は、『Oracle Rdb7 SQL Reference Manual』の DECLARE ALIAS 文を参照してください。

読取り保護操作を実行するトランザクションは、EMPLOYEES テーブルを共有します。他の更新トランザクションのアクセスは許可されません。例 3-24（3-66）に、読取り保護トランザクションを宣言する方法を示します。

例 3-24 読取り保護モードによるトランザクションの開始

```
SQL> ATTACH 'FILENAME mf_personnel';
SQL> SET TRANSACTION READ WRITE RESERVING
  1> EMPLOYEES FOR PROTECTED READ;
```

例 3-25（3-66）に示す更新トランザクションは、EMPLOYEES テーブルにアクセスし、他の検索トランザクションとアクセスを共有します。複数の更新トランザクションの同時実行は許可されません。

例 3-25 書込み保護モードによるトランザクションの開始

```
SQL> ATTACH 'FILENAME mf_personnel';
SQL> SET TRANSACTION READ WRITE RESERVING
  1> EMPLOYEES FOR PROTECTED WRITE;
```

例 3-26（3-67）に示す排他的共有モードでは、データベース・リソースへの一度に 1 トランザクションのみのアクセスが許可されます。排他的読取りモードでは EMPLOYEES テーブルのデータの読取りだけを許可されますが、排他的書込みモードでは EMPLOYEES テーブルのデータの挿入、更新、削除を許可されます。

例 3-26 排他的読取りモードまたは排他的書込みモードによるトランザクションの開始

```
SQL> ATTACH 'FILENAME mf_personnel';
SQL> SET TRANSACTION READ WRITE RESERVING
  1> EMPLOYEES FOR EXCLUSIVE READ;
  2> COMMIT;
SQL>
SQL> SET TRANSACTION READ WRITE RESERVING
  1> EMPLOYEES FOR EXCLUSIVE WRITE;
```

最大の同時アクセスを保証するには、データベース・リソースを要求するときに、最も互換性の高いロックを選択します。たとえば、インデックスに競合の問題がある非常に短いトランザクションでは、書込み保護を使用すると同時ユーザーが存在しても共有書込みより高速になる場合があります。互換性のあるロックは、同じデータベースに関して他のロックとの共存ができます。表 3-9（3-67）に、現行トランザクションと他のトランザクションが指定できるアクセス・モードとの互換性を示します。

表 3-9 現行トランザクションと他のトランザクションが指定できるアクセス・モードとの互換性

要求するモード	現行ロックのモード				
ロック	SR	SW	PR	PW	EX
SR	あり	あり	あり	あり	なし
SW	あり	あり	なし	なし	なし
PR	あり	なし	あり	なし	なし
PW	あり	なし	なし	なし	なし
EX	なし	なし	なし	なし	なし

ロック・モードのキー

- SR - 共有読取り（Shared Read）。同時読取り（CR: Concurrent Read）と呼ぶこともあります。
- SW - 共有書込み（Shared Write）。同時書込み（CW: Concurrent Write）と呼ぶこともあります。
- PR - 読取り保護（Protected Read）

- PW - 書込み保護（Protected Write）
- EX - 排他（Exclusive）
- あり - ロック間の互換性があります。
- なし - ロック間の互換性がありません。

場合によっては、他のトランザクションがテーブルの検索や更新を行わないようにする必要があります。SQL の SET TRANSACTION 文で排他モードを指定すると、Oracle Rdb ではテーブル全体がロックされます。テーブルが複合記憶領域にある場合、排他的アクセスのトランザクションでは次の状況で別のトランザクションがテーブルにアクセスすることはできません。

- テーブルのハッシュ・インデックスが定義されている場合
- 他のトランザクション（1 つ以上）が読取り / 書込みの場合
- 他のトランザクション（1 つ以上）がクエリーでハッシュ・インデックスを使用する場合

これは、同じ記憶領域内のテーブルが同じシステム・レコードを共有するためです。Oracle Rdb がテーブルのハッシュ・バケットを排他モードで更新すると、システム・レコードがロックされるので、他のテーブルへのアクセスが阻止されます。排他モードの更新が必要なテーブルは、別々の領域に配置してください。

次の指針に従うと、他のユーザーによるデータベース内のテーブルと行へのアクセスを許可し、ロックの互換性を保証できます。

- レコード・ストリームを正確なレコード選択の式に制限することで、必要な行のみを指定します。
- 1 つのトランザクション内ではクエリーの数をできるだけ減らします。
- SQL の SET TRANSACTION 文で互換性のあるモードを指定し、他のユーザーも同じリソースに同時にアクセスできるようにします。

Oracle Rdb がロックによってデータベース・リソースを保護する方法の詳細は、3.8.5 項（3-76）を参照してください。

3.8.3.1 互換性のない共有モードとロックのタイプ

Oracle Rdb では、排他的書込み共有モードとデフォルトの読取りロック・タイプを使用する読取り専用トランザクション・タイプは互換性がないと見なされます。排他的書込み共有モードではスナップショット・ファイルへの書込みを行わないためです。排他的書込みモードでトランザクションを開始すると、排他的書込みトランザクションがコミット操作（COMMIT 文による）かロールバック（ROLLBACK 文による）操作を実行するときに後続の読取り専用トランザクションが待機したり、異常終了したりします。読取り専用トランザクションに WAIT 修飾子があると、Oracle Rdb はリソースがロックされていることを示す

メッセージを返し、NO_WAIT 修飾子に変換し、さらに記憶領域がスナップショット・ファイルの準備をできないことを示すメッセージを返します（例 3-27（3-69））。

例 3-27 共有モードとロック・タイプに互換性がない場合のエラー・メッセージ

```
SQL> ATTACH 'FILENAME mf_personnel';
SQL> SET TRANSACTION READ ONLY WAIT;
SQL> SELECT * FROM EMPLOYEES;
%RDB-E-LOCK_CONFLICT, request failed due to locked resource
-RDMS-F-CANTSNAP, can't ready storage area $DUA0:[ORION]EMPIDS_LOW.RDA;1
for snapshots
```

したがって、現行トランザクションが読取り専用トランザクションのユーザーはロールバックする必要があります。

トランザクションでデータベースの特定の領域への排他的アクセスが必要な場合は、スナップショット・ファイルにアクセスする他の読取り専用トランザクションに排他モードがおよぼす影響について認識する必要があります。

3.8.3.2 キャリーオーバー・ロックと [NO]WAIT オプション

キャリーオーバー・ロックの最適化によって、トランザクションはコミットの際に発生する一部の論理領域と物理領域のロック要求のオーバーヘッドを回避できます。この項では、キャリーオーバー・ロックの最適化が機能する方法の概要を説明し、WAIT トランザクションと NOWAIT トランザクションへの個々の影響を説明します。

現行トランザクションによる領域ロック要求は、**アクティブ**・ロックと呼ばれます。コミットの際に、Oracle Rdb は領域ロックを降格しようとします。コミットの際に降格しない領域ロックは、**キャリーオーバー**・ロックと呼ばれます。

データベースに接続している間に、プロセスにはアクティブなロック（現行トランザクションで使用するロック）とキャリーオーバー・ロック（以前のトランザクションで要求したロックが降格していない）が存在する場合があります。トランザクションで現在キャリーオーバーとマークされたロックが必要な場合、アクティブなロックに変更することでロックを再利用します。したがって、同じロックがアクティブからキャリーオーバーへ、またアクティブへと何度も変換されるので、ロック要求と降格の費用を節減できます。これで、プロセスが同じ領域セットに繰り返しアクセスする場合は、ロック要求の数を大幅に削減できます。

WAIT トランザクション

WAIT トランザクションが領域のロックを要求すると、Oracle Rdb はプロセス A が領域 X をロックし、プロセス B が同じ領域にアクセスしようとする場合に、必ず次の 2 つのケースを区別します。

- プロセス A が領域 X のキャリーオーバー・ロックを実行している場合、A は要求あり次第ロックを明け渡し、プロセス B がロックを実行して処理を続けます。

- プロセス A が領域 X のアクティブ・ロックを実行していると、A はそのトランザクションが完了するまでロックを明け渡すことができません。この場合、Oracle Rdb ではフラグを設定し、A のトランザクションがコミットするときにこのロックを降格して B がロックできるようにする必要があることを示します。プロセス B は要求あり次第ロックを行うことはできないので、待機する必要があります。

WAIT トランザクションでは、キャリーオーバー・ロックの最適化に伴うロック回数の削減によって、ブロッキング AST が増加する場合があります。ブロッキング AST の増加は、ロックに関する Performance Monitor の各画面で確認できます。

キャリーオーバー・ロックの最適化は、各トランザクションがすべてのパーティションのデータにランダムにアクセスするのでなく独自のデータ・セットにアクセスするように設計されているので、競合が増える場合に適切に機能します。たとえば、EMPLOYEES テーブルを 3 つの領域に分割する EMPLOYEE_ID 列について考察します。EMPLOYEES テーブルにアクセスするアプリケーションは、トランザクションが任意の領域をランダムに選択するのでなく、特定の領域またはデータ・セットにアクセスするように設計する必要があります。さらに、キャリーオーバー・ロックの最適化は各トランザクションが同じ領域またはデータ・セットに繰り返しアクセスする場合に最適な機能を果たします。この点で、Oracle Rdb で使用できるパーティション化と配置の機能が役に立ちます。

NOWAIT トランザクション

NOWAIT トランザクションは、ロックを待機しません。NOWAIT トランザクションが要求するロックがただちに許可されない場合はエラー・メッセージが表示され、トランザクションは異常終了します。キャリーオーバー・ロック最適化の一環として、NOWAIT トランザクションでは NOWAIT ロックを要求、取得および保持します。これで、データベースにアクセスする他のプロセスに NOWAIT トランザクションが存在することを示し、キャリーオーバー・ロックが解放されます。キャリーオーバー・ロックが解放されないと、NOWAIT トランザクションは WAIT トランザクションのプロセスがデータベースの接続を解除するまでその WAIT トランザクションがキャリーオーバー・ロックを保持する領域にはアクセスできません。

ただし、別のトランザクションがロックを保持している場合は、NOWAIT トランザクションが NOWAIT ロックを取得する際に遅延が発生します。この場合は、次に示す Performance Monitor のストール・メッセージが表示されることがあります。

```
waiting for NOWAIT signal (CW)
```

NOWAIT トランザクションの実行が著しく遅い場合は、SQL の CREATE DATABASE 文か SQL の ALTER DATABASE 文 CARRY OVER LOCKS ARE [ENABLED | DISABLED] 句を使用してキャリーオーバー・ロックの最適化を無効にします。キャリーオーバー・ロックはデフォルトで有効です。例 3-28（3-71）に、デフォルトで有効なキャリーオーバー・ロックを無効にする方法を示します。

例 3-28 キャリーオーバー・ロックの無効化

```
SQL> ALTER DATABASE FILENAME test1
  1> CARRY OVER LOCKS ARE DISABLED;
```

キャリーオーバー・ロックが有効か無効かは、Performance Monitor の「Lock Information」画面で確認できます。次に例を示します。

```
Node: TRIXIE          Oracle Rdb V7.0-00 Performance Monitor 27-JUN-1996 13:57:26
Rate: 3.00 Seconds              Lock Information            Elapsed: 00:06:55.94
Page: 1 of 1       RDBVMS_USER1:[LOGAN.V70]MF_PERSONNEL.RDB;1       Mode: Online
------------------------------------------------------------------------------
Adjustable record locking granularity is enabled
- Fanout factor 1 is 10 (10 pages)
- Fanout factor 2 is 10 (100 pages)
- Fanout factor 3 is 10 (1000 pages)
Carryover lock optimization is enabled
Lock Tree Partitioning is disabled
Lock timeout is disabled
------------------------------------------------------------------------------
Exit Help Menu Options Refresh Set_rate Write !
```

キャリーオーバー・ロックの最適化を無効にすると、パフォーマンスが若干低下することがあります。これは、複数のトランザクション全体で領域とロックを保持するのでなく、個々のトランザクションが領域と最高レベルの ALG ロックを取得したり解放したりするためです。

3.8.3.3 テーブルの更新キャリーオーバー・ロックの最適化

標準的な環境では、プロセスが更新トランザクションを開始すると、まず 1 行を読み取り、次にその行を更新します。この場合、Oracle Rdb はまずプロセスに論理領域の CR（同時読取り）モードによるロックを許可し、後でロックを CU（同時更新）モードにアップグレードします。プロセスが別の更新トランザクションを開始する場合も、同じ状況が発生することがあります。多くのロック操作では、CR モードから CU モードに変更し、また元に戻すように要求されることがあります。

プロセスの RDMS$AUTO_READY 論理名または RDB_AUTO_READY 構成パラメータを定義し、プロセスのテーブル・レベルの更新キャリーオーバー・ロックを有効にします。RDMS$AUTO_READY 論理名か RDB_AUTO_READY 構成パラメータを定義すると、プロセスが論理領域のロックを CR モードで要求したときに論理領域に対する CU モードのキャリーオーバー・ロックがすでに保持されている場合は、CU モードのロックを取得します。この論理領域に関するキャリーオーバー・ロックの最適化によって、多くの更新トランザクションに関するロックのオーバーヘッドを軽減します。

OpenVMS VAX OpenVMS Alpha

例 3-29（3-72）に、プロセスに関するテーブル・レベルの更新キャリーオーバー・ロックを有効にする方法を示します。ただし、キャリーオーバー・ロックが有効になるのは

RDMS$AUTO_READY 論理名を 1 に定義した場合だけです。それ以外の場合、論理名の定義によってテーブル・レベルの更新キャリーオーバー・ロックが有効になることはありません。

例 3-29 プロセスに関するテーブル・レベルの更新キャリーオーバー・ロックの有効化

```
$ DEFINE RDMS$AUTO_READY 1
```

プロセスに関するテーブル・レベルの更新キャリーオーバー・ロックを無効にするには、RDMS$AUTO_READY 論理名の割当てを削除します。◆

プロセスが更新キャリーオーバー・ロックの最適化による恩恵を被るのは、データベースを更新している場合だけです。プロセスに関してテーブル・レベルの更新キャリーオーバー・ロックを有効にすると、プロセスが論理領域に対する CU モードのキャリーオーバー・ロックを保持するテーブルで新しい更新トランザクションを開始するときに、論理領域に関する CU モードのロックを取得するという効果があります。すなわち、トランザクション開始時に CU モードのロックを取得するので、プロセスはロックをまずダウングレードしてから後でアップグレードするオーバーヘッドが回避できます。プロセスが更新を行わない場合は、論理領域のロックをより低い（CR）モードで取得すれば十分です。

RDMS$AUTO_READY 論理名と RDB_AUTO_READY 構成パラメータは、大量の更新集中型のトランザクション処理環境において、トランザクションごとにロックの変換が何度も発生していることが Performance Monitor によって示される場合に使用してください。

テーブル・レベルのキャリーオーバー・ロックを有効にしたプロセスでは、そのプロセスが PROTECTED READ モードか PROTECTED WRITE モードでテーブルを予約する場合、またはテーブルの順次スキャンを実行する場合に、並行性の問題の原因になることがあります。

3.8.3.4 "lock conflict on freeze lock" エラーの説明

場合によっては、NOWAIT トランザクションで "lock conflict on freeze lock" というエラー・メッセージが表示されることがあります。次に、この危険性の高い例について考察します。

プロセス A	プロセス B
Holds lock on resource 1.	
	Requests lock on resource 1 and is waiting.
Modifies resource 1.	
Process A is terminated prematurely.	
Operating system releases all locks held by A.	
	B is granted the lock on resource 1 and may see uncommitted changes made by A.

Oracle Rdb がクラスタ内で動作する場合は、プロセス（プロセス A）が終了しても、エラーを検出して障害プロセスのトランザクションをリカバリするまでは（必要な場合）、データベース・モニターが他のアプリケーション・プロセス（プロセス B）にロックを許可しないという保証が必要です。

"freeze プロトコル " を使用すると、別のアプリケーション・プロセスから新しいロック要求があっても、終了したプロセスに必要なクリーンアップ（リカバリ）操作をモニターが実行できるようになるまでは許可されないことが保証されます。この例では、freeze プロトコルによって、終了したプロセス（プロセス A）のリカバリ操作が完了するまでプロセス B のロック要求が許可されないことが保証されます。

次に、"freeze プロトコル " アルゴリズムがどう機能するのかを示します。ただし、ここでは詳細を省略して簡単に説明します。

個々のユーザー・プロセスは、データベースに接続すると同時書込み（CW）モードの FREEZE ロックを取得します。

データベースのリカバリを要する可能性がある場合は（プロセスが異常終了した場合やクラスタ内のノードに障害が発生した場合など）、必ずデータベース・リカバリ（DBR）プロセスによって読取り保護（PR）モードの FREEZE ロックが要求されます。DBR プロセスは非互換モードの FREEZE ロックを要求するので、CW モードの FREEZE ロックを保持するユーザー・プロセスごとにブロッキング AST が生成されます。各プロセスが FREEZE ロックを解放すると、DBR プロセスは FREEZE ロックを PR モードで取得し、リカバリ操作を開始します。

ユーザー・プロセスがリソースのロック要求を許可されるたびに、Oracle Rdb はそのプロセスが CW モードの FREEZE ロックを保持しているかどうかも確認します。プロセスが CW モードの FREEZE ロックを保持しない場合、Oracle Rdb はモニターがデータベースをリカバリして一貫性のある状態に戻そうとしていることを察知します。この場合は、DBR プロセスがリソースのロックを取得してデータベースをリカバリできるように、リソースのロック要求を許可されたプロセスがそのロックを解放する必要があります。

Oracle Rdb は、プロセスがリソースに対して許可されたロックを解放し、（プロセスのロック要求が WAIT 要求の場合）プロセスの FREEZE ロックを CW モードで要求します。これが許可されてから、Oracle Rdb はプロセスのオリジナル・ロックを再発行します。

プロセスのロック要求が NOWAIT ロックの場合、Oracle Rdb は "lock conflict on freeze" メッセージを発行し、リカバリが進行中の（可能性がある）ためにロック要求が許可されないことを示します。次のいずれかのイベントによって（おそらく他のイベントによっても）、エラー・メッセージ "lock conflict on freeze" が表示されます。

- ロック要求が NOWAIT 要求でリカバリが行われている場合（モニター・ログの DBR プロセスが示す）
- ユーザーがデータベースにアクセスする間にノードに障害が発生した場合
- モニター・ログに、"cluster recovery completed successfully"、"received request from remote node to join..." などのイベントが記録された場合。これは、ノード上で新しいデータベース・アクティビティが発生しているか、ノード上でデータベースにアクセスするユーザーが他にいないことを示します。このようなイベントが発生した場合、モニ

ターはこの推移によってデータベースが一貫性のない状態にならないことを保証する必要があります。

3.8.3.5 バッチ更新トランザクション

バッチ更新トランザクションは、大規模な初期ロード・トランザクションのオーバーヘッドを軽減したい場合に有効です。ただし、バッチ更新トランザクションには問題が発生した場合にいくつかのリスクがあるので、オラクル社ではこのような初期ロード操作には読取り / 書込みの排他モードの使用をお薦めします。バッチ更新トランザクションでは .ruj ファイルが作成されないので、トランザクションに障害が発生した場合にロール・バックすることができません。したがって、データベースが破損し、最新のバックアップ・ファイルからデータベースをリストアするか最初から作り直す必要があります。バッチ更新トランザクションを使用してテーブルをロードする場合は、バッチ更新トランザクションで初期ロード操作を行う前に必ずデータベースをバックアップしてください。排他的共有モードを使用すると、.ruj ファイルが作成されるので、トランザクションに障害が発生した場合はトランザクションをロール・バックできます。バッチ更新トランザクションの使用方法は、『Oracle Rdb7 SQL プログラミングのガイド』を参照してください。

3.8.3.6 カーソルの更新ロッキング

カーソルを使用してレコード・ストリームを選択し、行の読取り（と更新）を行う場合、Oracle Rdb はフェッチされた各行が更新されるかどうかを認識できません。デフォルトの Oracle Rdb の動作は、指定された行の読取りをロックしてから、行が更新される場合は書込みのロックにアップグレードします。現行バージョンの Oracle Rdb では、フェッチされた行のすべてまたはほとんどが更新されることが事前にわかっていれば、初期読取り操作の際により限定的なロックを適用できます。これで、ロックのオーバーヘッドが軽減され、デッドロックの発生が削減されます。

更新カーソルを定義するには、DECLARE CURSOR 文の UPDATE ONLY 句を使用します。次の例に一般的な文を示します。構文の詳細は『Oracle Rdb7 SQL Reference Manual』を参照してください。

```
DECLARE cursor_name UPDATE ONLY CURSOR FOR select-expr
```

3.8.4 トランザクション・スコープ

すべてのデータベース・アクティビティは、トランザクション内で制御されます。トランザクションを開始する際には、そのトランザクションがアクティブな間に、他のユーザーがデータに対して実行できるデータベース・アクセスの種類を指定します。使用するテーブルへのすべてのアクセスを制限することも、同じデータを他のユーザーと共有することもできます。トランザクションがどのアクセス・モードを指定しても、他のアクティブなトランザクションとともに、データ・リソースへの制限を強化します。通常は、他のトランザクションが行を解放するまで待機しないと行を検索できません。Oracle Rdb では、ユーザーはレコード・ストリームから 1 行または複数の行をフェッチできます。検索する行が多いほど

(特に更新の場合)、別のユーザーと競合する可能性は高くなります。トランザクションを終了すると、検索した行に関するすべてのロックを解放します。

他のユーザーとデータベース・ロックが競合する可能性を低下するには、トランザクションの持続時間をできるだけ短くし、トランザクションの中でデータを最大限に共有する必要があります。行を検索してユーザーの端末に特定の列を表示する場合は、共有モードを指定して他のユーザーが同じ行を読み取るのを許可します。SQL の SET TRANSACTION 文で READ ONLY を指定すると、検索する行へのアクセスを共有できます。行のロックを保持するトランザクション内でユーザーに端末入力を要求するトランザクションは記述しないでください。

データベースを更新する場合は、読取り / 書込みトランザクションで共有書込みを指定します。行を更新すると同時に、トランザクションを終了します。こうして、ロックしたリソースをシステムに解放し、別のユーザーによる行の読取りや更新を許可します。読取り専用モードで行を検索してから行の更新を決定した場合は、読取り / 書込みを指定してデータベース内の行を変更します。行を削除する場合、削除した行の dbkey を Oracle Rdb でいつ再利用できるかは dbkey スコープによって決まります。DBKEY SCOPE IS TRANSACTION を指定すると（デフォルト)、Oracle Rdb は元の行を削除したトランザクションが COMMIT 文で完了するまで、削除した行の dbkey を別の行の保存には使用できません。データベース・ユーザーのだれかが DBKEY SCOPE IS ATTACH を指定すると、Oracle Rdb は削除した行の dbkey を別の行の保存には使用できません。

dbkey スコープは次のように指定します。

- すべてのアクティブ・ユーザーが DBKEY SCOPE IS TRANSACTION 句を使用すると、すべてのユーザーの dbkey スコープが TRANSACTION になります。
- ただ 1 人のアクティブ・ユーザーが DBKEY SCOPE IS ATTACH を使用しても、すべてのユーザーの dbkey スコープが ATTACH になります。

トランザクションの持続時間をできるだけ短くすると、トランザクション内のデータベース・アクティビティが最小化され、最適なレベルの同時アクセスを保持するのに役立ちます。システム障害が発生した場合に、更新トランザクションが短ければ迅速にリカバリでき、最後にコミットしたトランザクションまでデータベースを復元できます。コミットされていないトランザクションが長いと、システム障害の後処理が長くなります。ロールバックした内容の再入力が必要になります。.ruj ファイルのサイズは更新トランザクションのスコープに伴って増大するので、.ruj ファイルに必要なディスク領域も増大する恐れがあります。

表 3-10（3-76）に、トランザクション・スコープによるデータベース・ユーザー、ファイル、パフォーマンスへの影響を要約します。

表 3-10 データベース・トランザクションの長さによる影響

更新トランザクションのエクステント	影響	
排他的共有モードで長い	メリット	更新が迅速 アクセスの競合がない 最小限のロック
	デメリット	並行性の低下 システム障害のリカバリが遅い
共有共有モードで長い	メリット	並行性の向上
	デメリット	多くのロックを要する 競合が多いので更新に時間がかかる
短い	メリット	データベースが最新 .ruj ファイルのサイズと必要なストレージが小さい システム障害のリカバリが速い 並行性の向上
	デメリット	トランザクション数が多く、ロックマネージャのオーバーヘッドが大きい

3.8.5 Adjustable Lock Granularity（ALG）

Adjustable Lock Granularity（ALG）は、トランザクションの間に特定の論理領域内のページに対して最小限のロックで最大限の同時アクセスを確保するために Oracle Rdb が使用するメカニズムです。ALG は行のグループ（複数のインデックス・ノードまたはデータ行）をロックします。このとき、同じトランザクションで後からアクセスが必要になる行も、ロックしたグループに含まれることが期待されます。

ALG は論理的な逆ツリー構造に基づいており、データベース・リソースに関する様々なレベルのロックを保持します。Oracle Rdb では、ページ範囲のレベルのデフォルトが 3 で、各レベルのファンアウト・ファクタは 10 です。ADJUSTABLE LOCK GRANULARITY 句の COUNT パラメータを使用してページ・レベルの値を指定すると、ロックの粒度をさらに厳密に制御できます。COUNT パラメータの値は 1 ～ 8 です。8 は次のように解釈されます。

- レベル 10。逆ツリーの最上位（ルート）であり、論理領域内のすべての行が含まれます。
- レベル 9。論理領域内の連続する 1,000,000,000 ページの行が含まれます。
- レベル 8。論理領域内の連続する 100,000,000 ページの行が含まれます。
- レベル 7。論理領域内の連続する 10,000,000 ページの行が含まれます。
- レベル 6。論理領域内の連続する 1,000,000 ページの行が含まれます。

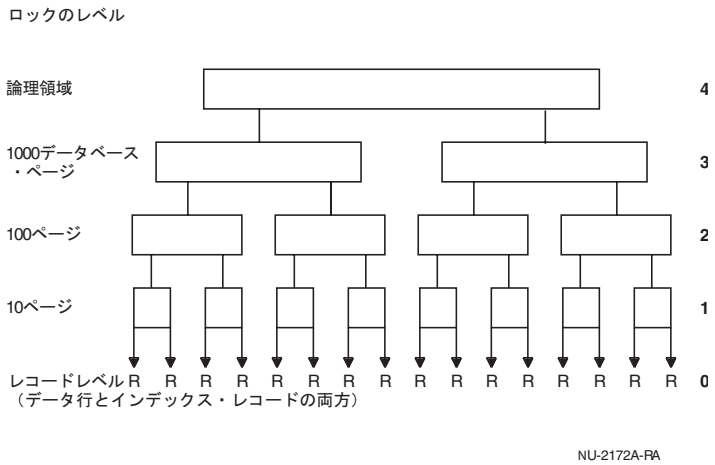
- レベル 5。論理領域内の連続する 100,000 ページの行が含まれます。
- レベル 4。論理領域内の連続する 10,000 ページの行が含まれます。
- レベル 3。論理領域内の連続する 1000 ページの行が含まれます。
- レベル 2。論理領域内の連続する 100 ページの行が含まれます。
- レベル 1。論理領域内の連続する 10 ページの行が含まれます。
- レベル 0。逆ツリーの最下位（リーフ）であり、行レベルを表します。

データベースのページの競合が大きい（同じ領域に多くのユーザーが同時にアクセスする）場合は、COUNT 値の指定を下げることを検討します。少数のデータベース・ユーザーがクエリーを実行し、広く分散した多くの行にアクセスする場合は、COUNT 値の指定を上げるとロックが少なくなります。

ADJUSTABLE LOCK GRANULARITY IS DISABLED を指定すると、Oracle Rdb はデータベース行が要求されるたびにその行のロックを要求します。Oracle Rdb では、起動時にデフォルトの COUNT 値で ALG を有効化し、それから CPU 時間に問題があれば COUNT 値の調整や ALG の無効化を行い、パフォーマンスが改善されたかどうかを判断することをお薦めします。

図 3-1（3-77）に、デフォルトの ALG ツリー構造を示します。

図 3-1 ALG のレベル



ページ・レベルとしてデフォルト値の 3 を使用すると、初めはすべてのトランザクションがレベル 4（論理領域）の強力なロックを要求します。競合がない（同じ論理領域内の行へのアクセスを待機するトランザクションが他にない）場合、トランザクションには論理領域内

のすべての行に関する唯一のロックが必要です。これは、レベルの高い強力なロックによってその支配下にある低レベルのオブジェクトがすべて暗黙的にロックされるためです。

別のトランザクションで同じ論理領域の行へのアクセスが必要な場合は、最初のトランザクションによるレベル 4 の強力なロックが段階的に縮小され、該当するページ範囲に対してレベル 3 の強力なロックが要求されます。2 番目のトランザクションもレベル 4 の緩やかなロックを取得し、該当するページ範囲に対してレベル 3 の強力なロックを取得します。レベル 3 で競合する場合は、最初のトランザクションの強力なレベル 3 のロックが緩やかなレベル 3 のロックに段階的に縮小され、該当するページ範囲に対する強力なレベル 2 のロックが要求されます。2 番目のトランザクションについても、同じ手順を実行します。上記のようなロックの取得と段階的縮小の組合せは、ALG ツリーのリーフに向かって競合がなくなるまで続きます。2 つのトランザクションが同じ行にアクセスしようとすると、行レベルの競合が発生します。この場合、2 番目のトランザクションは最初のトランザクションが終了してロックを解放するまで待機する必要があります。

必要なロックの数は、論理領域内の行にアクセスするトランザクションの数と、ALG ツリー内で発生した競合の数によって変わります。デフォルトでは、最悪の場合、トランザクションは 4、3、2、1 とレベルを下げ、最終的にレベル 0 で強力なロックを取得する必要があります。したがって、1 行にアクセスするのに合計で 5 回ロックを要求することになります。このように最悪のケースは、近接する行で激しい競合がある場合にのみ発生します。

このメカニズムは、多くのトランザクションが様々なページのグループに同時にアクセスしなければならない場合に有効です。ALG を有効にすると、トランザクションが競合のないページ範囲のデータ行にアクセスするように設計されたアプリケーションではロック・アクティビティを削減できます。

一般に、互いに近接する多くの行にアクセスするトランザクションでは、ALG を有効にする必要があります。ALG はトランザクションで可能な最高レベルのロックを取得するので、トランザクションごとに最小限のロックで行にアクセスできます。

トランザクションでアクセスする論理領域内の行が少ない場合は、ALG を無効にする必要があります。この場合は、Oracle Rdb がただちに最低レベルのロックを実行する方が効率的です。ALG が有効で、低レベルでページの競合がある場合は、トランザクションでアクセスするのが 1 行のみでも、ALG は ALG ツリーのすべての中間レベルでロックを取得する必要があることを忘れないでください。

トランザクションで数百万行へのアクセスが必要な場合は、そのトランザクションが保護モードでテーブルをロックするように設定する必要があります。保護モードのロックによって、行にアクセスするたびに暗黙的または明示的なロックを実行するオーバーヘッドが軽減されます。これで、仮想メモリーをかなり削減できます。換言すれば、トランザクションでテーブル内のほとんどの行にアクセスする場合は、保護モードでテーブルを予約する必要があります。

注意：ALG は、個々のトランザクションでなく、データベース全体に関して有効化または無効化されます。データベースに対して実行するすべてのトランザクションの一般的な特性を確認してから、ALG を有効にするか無効にするかを決定してください。

データベース作成時には、デフォルトで ALG が有効になっています。ALG を無効にする場合は、SQL の ALTER DATABASE 文の ADJUSTABLE LOCK GRANULARITY IS DISABLED 句を使用します。ALG の有効化が必要な場合と無効化が必要な場合の詳細は、8.4 項（8-43）を参照してください。

Performance Monitor の「Stall Messages」画面には、ALG ツリー内でロックが存在する場所が示されることがあります。たとえば、ストール・メッセージ「waiting for record 5:0:-4」はデータベース・システムが ALG ロックを取得しようとしていることを示しており、次のように解釈します。

- 最初の数（ここでは 5）は論理領域番号を示します。
- 2 番目の数（ここでは 0）は、このロックでロックされるページ範囲の開始ページを示します。
- 負の数（ここでは -4）は ALG ツリーのレベルを示しており、論理領域のロックを示す -10 から 10 ページの範囲のロックを示す -1 までを指定できます。

「Stall Messages」画面の使用方法の詳細は、3.2.1.3 項（3-9）を参照してください。

3.8.6 ページ・レベルのロックと行レベルのロックの選択

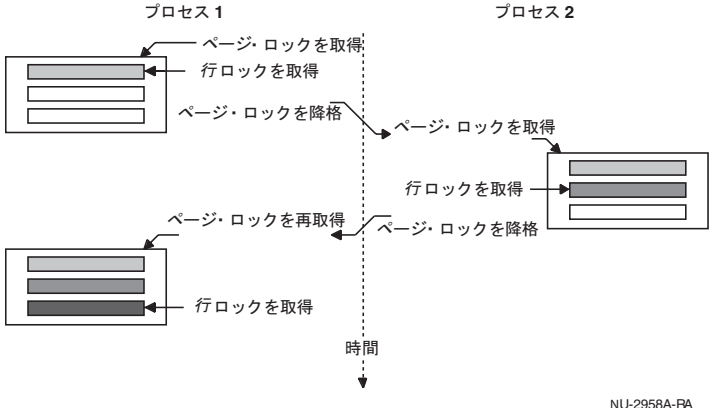
Oracle Rdb では、行レベルのロックによって論理的な一貫性を提供し、ページ・レベルのロックによって相互排他を提供します。ページ・レベルと行レベルのロック・メカニズムは互いに独立して機能します。

図 3-2（3-80）に、同一のデータ・ページの別々のレコードにアクセスする 2 つのプロセスを示します。行レベルのロックによって論理的な一貫性が提供され、ページ・レベルのロックによって変更が 1 つずつ順番に実行されることが保証されます。

図 3-2（3-80）では、プロセス 1 がページ・ロックとそのページの 1 行に対する行ロックを取得しています。プロセス 2 がそのページの別の行にアクセスする場合、プロセス 1 はページ・ロックを解放し、プロセス 2 がページロックとそのページの 1 行に対する行ロックを取得します。プロセス 1 がそのページの別の行にアクセスする場合、プロセス 2 はページ・ロックを解放し、プロセス 1 がページロックとそのページの 1 行に対する行ロックを取得します。プロセスが行ロックを取得すると、トランザクションが終了するまでその行ロックを保持します（このプロセスは、別のプロセスがそのページのページ・ロックを取得しても引き続き行ロックを保持します）。Oracle Rdb で更新が許可されるのは、プロセスが該当するページのページ・ロックを保持する場合だけです（トランザクションは分離レベル

SERIALIZABLE で実行されるものとします)。ページ・ロックを保持するのは 1 度に 1 プロセスだけなので、同一のページの行を変更できるのも 1 度に 1 プロセスだけになります。

図 3-2 2 つのプロセスが同一のデータ・ページの別々のレコードにアクセスする場合のページ・レベルと行レベルのロック



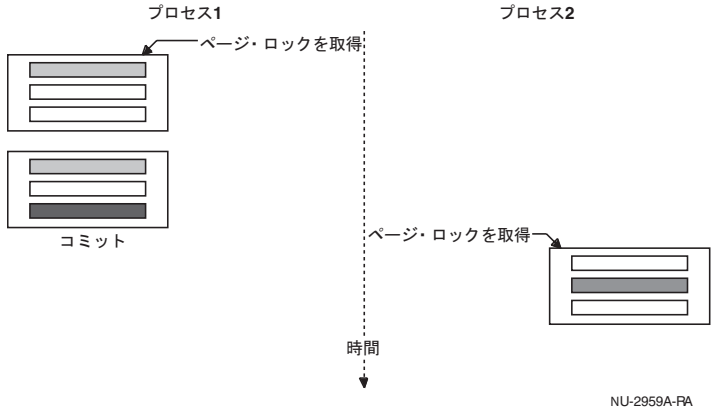
SQL の CREATE DATABASE 文と ALTER DATABASE 文の LOCKING IS PAGE LEVEL 句を使用すると、データベース内の 1 つ以上の記憶領域またはすべての記憶領域にページ・レベルのロックのみを適用するように指定できます。上記の文の構文は、『Oracle Rdb7 SQL Reference Manual』を参照してください。記憶領域に関するページ・レベルのロックが有効な場合は、その記憶領域にアクセスするトランザクションはページ・レベルのロックのみを保持し、行レベルのロックは要求しません。ページ・レベルのロックによって論理的な一貫性と相互の除外の両方が提供されます。

記憶領域に関するページ・レベルのロックが有効な場合は、その領域内の行にアクセスするプロセスはアクセスするページごとのページ・ロックを許可されますが、ページ上のアクセスする行に関する行レベルのロックを取得する必要はありません。これで、ロックのオーバーヘッドが軽減されます。

LOCKING IS PAGE LEVEL 句を指定すると、ページのロックはトランザクションが終了するまで保持されるので、そのページの行にアクセスする他のトランザクションをブロックすることがあります。したがって、LOCKING IS PAGE LEVEL 句を指定すると、ロック操作は削減されますが、並行性も低下することがあります。図 3-3（3-81）ではページ・レベルのロックが有効化されており、図 3-2（3-80）に示す 2 つのプロセスは同じデータ・ページの同じ行にアクセスしています。ページ・レベルのロックが有効な場合は、ページ・ロックを取得するプロセスは COMMIT 文か ROLLBACK 文によってその現行トランザクションを終了するまでロックを保持することが許可されます。ページ・ロックを保持するプロセスがその現行トランザクションを完了するまで、他のユーザーはこのページからブロックされます。図 3-3（3-81）では図 3-2（3-80）よりも使用するロックの数は少なくなりますが、各プロセスがページ・ロックを保持すると残りのプロセスをブロックするので並行性も低下しま

す。多くのプロセスがそのページにアクセスする場合は、そのすべてのプロセスのストール時間が増大します。

図 3-3 ページ・ロックが有効で 2 つのプロセスが同一のデータ・ページの別々のレコードにアクセスする場合の並行性の低下



ページ・レベルのロックの有効化は、個々のアプリケーション・プロセスが同時に同じデータ・ページにはアクセスしないパーティション・アプリケーションで使用した場合に大いに効果が期待されます。図 3-4（3-81）に、従業員に関する情報が地理的な場所に基づいて分割されている給与アプリケーションを示します。すなわち、東部の従業員に関する情報は EAST 記憶領域に、西部の従業員に関する情報は WEST 記憶領域に格納されています。地理的な場所ごとに別々のアプリケーション・プロセスを使用して従業員情報を更新する場合、複数のアプリケーション・プロセスで同時に同じデータ・ページを要求することはほとんどありません。このような状況では、ページ・レベルのロックを有効にすると行ロックのオーバーヘッドがなくなるので、並行性を損なうことなくパフォーマンスを改善できます。

図 3-4 パーティション・アプリケーションのページ・ロックによる効果

ページ・レベルのロックの有効化は、次の特性をもつアプリケーションにも効果的な場合があります。

- リソースの競合が限定的
  ページ・レベルのロックを有効にすると、特定のページとそのページの行に関する 1 ページのロックを迅速に取得できます。
- トランザクションの持続時間が短い
  ページ・レベルのロックを有効にすると、プロセスは 1 つのトランザクションを終了するまでページのロックを保持します。トランザクションが短い場合、ページが別のプロセスですでにロックされていても、競合するプロセスの待機はほんのわずかな時間で済みます。

ページ・レベルのロックの有効化は、データ・ページのすべての行がそのページ上でクラスタ化されている記憶領域にアクセスするアプリケーションにも有効です。これは、そのハッシュ・インデックス・キーが同じ値であるためです。たとえば、行グループがクラスタ化（ハッシュ）されている場合、アプリケーションは 1 回のページ・ロックでページをフェッチしてすべての行を読取りおよび更新できます。行ごとに行レベルのロックを取得する必要はありません。ただし、このアプリケーションと同じデータ・ページに同時にアクセスするプロセスやアプリケーションが他にない場合以外は、ページ・レベルのロックの有効化による効果は得られません。ページ・レベルのロックが有効な記憶領域の同じデータ・ページに複数のプロセスが同時にアクセスしようとする場合は、各プロセスのトランザクション持続時間を非常に短くする必要があります。それ以外の場合は、各プロセスが絶えずストールしてページ・ロックを待機するのでパフォーマンスが低下します。

LOCKING IS PAGE LEVEL オプションは、必ずページ・レベルのロックを有効にした場合の影響について考察してから、慎重に使用する必要があります。記憶領域に対して LOCKING IS PAGE LEVEL オプションを使用すると、記憶領域全体に対してページ・レベルのロックが有効になります。多くのアプリケーションはテーブルなどの論理エンティティに関する操作を実行します。記憶領域内に複数のテーブルが存在する場合は、ページ・レベルのロックを有効にするときに特に注意する必要があります。記憶領域内に 1 つのテーブルの行だけが存在し、一度に 1 つのアプリケーションだけがテーブルにアクセスする場合は、ページ・レベルのロックの有効化を容易に決断できます。しかし、多くの場合、記憶領域の LOCKING IS PAGE LEVEL オプションを有効にすると、領域内のテーブルの数とその領域に同時にアクセスするアプリケーションの数が増えるにつれて、競合、遅延、デッドロックが増大します。

デフォルトは LOCKING IS ROW LEVEL です。1 つ以上の記憶領域に対して LOCKING IS ROW LEVEL オプションが有効な場合、その記憶領域にアクセスするトランザクションは行とページの両方のロックを使用します。一般に、LOCKING IS ROW LEVEL オプションは多くのトランザクションに適していますが、1 つ以上の記憶領域内の多くの行をロックするトランザクションや持続時間の長いトランザクションには特に有効です。

LOCKING IS PAGE LEVEL 句と LOCKING IS ROW LEVEL 句は、SQL の CREATE DATABASE 文と ALTER DATABASE 文に指定します。LOCKING IS PAGE LEVEL 句と LOCKING IS ROW LEVEL 句は、データベース・レベルに限らず、記憶領域レベルでも指定できます（CREATE STORAGE AREA 句または ALTER STORAGE AREA 句で使用）。

例 3-30（3-83）に、1 つの記憶領域（JOBS）に対してページ・レベルのロックを有効にする方法を示します。

例 3-30 記憶領域の LOCKING IS 設定の変更

```
SQL> -- Change the setting of the JOBS storage area from the default
SQL> -- setting of LOCKING IS ROW LEVEL to LOCKING IS PAGE LEVEL:
SQL> ALTER DATABASE FILENAME mf_personnel
  1> ALTER STORAGE AREA JOBS
  2>    LOCKING IS PAGE LEVEL;
SQL>
```

例 3-30（3-83）に、LOCKING IS 句を ALTER STORAGE AREA 句の一部として使用して、1 つの記憶領域に関するページ・レベルまたは行レベルのロックを指定する方法を示します。1 つ以上の記憶領域にページ・レベルか行レベルのロックを指定するには、記憶領域ごとに ALTER STORAGE AREA 句の一部として LOCKING IS 句を使用します。また、ALTER STORAGE AREA 句なしで LOCKING IS 句を使用すると、データベース内のすべての記憶領域に対してページ・レベルまたは行レベルのロックを有効にできます。これば簡単で便利な方法です。例 3-31（3-83）に、mf_personnel データベース内のすべての記憶領域に対してページ・レベルのロックを有効にする方法を示します。

例 3-31 データベース内のすべての記憶領域に関するページ・ロックの有効化

```
SQL> ALTER DATABASE FILENAME mf_personnel
  1> LOCKING IS PAGE LEVEL;
```

LOCKING IS PAGE LEVEL と LOCKING IS ROW LEVEL の設定はデータベース全体の属性ではなく記憶領域の属性であることを忘れないでください（記憶領域ごとに変更できます）。

Performance Monitor では、記憶領域ごとにロック・レベルの設定が表示されます。「Database Parameter Information」サブメニューで「Storage Area Information」画面を選択すると、ロック・レベルの設定を含む記憶領域の特性が表示されます。4.2.1.3 項（4-136）に、「Storage Area Information」画面の例を示します。

ページ・レベルまたは行レベルのロックを設定できるのは、データベースがオフラインの場合（他のユーザーがデータベースに接続していないとき）に限ります。

ページ・レベルのロックを有効にする場合には、次の制限があります。

- ページ・レベルのロックはシングルファイル・データベースでは無効です。シングルファイル・データベースでページ・レベルのロックを有効にしようとすると、RDB$_BAD_DPB_CONTENT エラーが返されます。
- SQL の CREATE DATABASE 文か ALTER DATABASE 文で RDB$SYSTEM 記憶領域にページ・レベルのロックを適用することはできません。ロック・プロトコルによってメタデータ・ユーザーがストールすることがあります。

- RDB$SYSTEM 記憶領域に明示的にページ・レベルのロックを指定することはできません。指定すると、Oracle Rdb は RDB$_BAD_DPB_CONTENT エラーを発行し、RDB$SYSTEM 記憶領域にはページ・レベルのロックを指定できないことを示します。

LOCKING IS PAGE LEVEL 機能を適切に使用するとトランザクションがロックを要求する回数は減りますが、この機能を使用することでクォータを調整すべきではありません。

3.8.7 リカバリ可能なラッチ

リカバリ可能なラッチは、ノード専用のロック操作で、競合の少ないリソースの排他的なロックが必要な場合に使用される Oracle Rdb ロックです。リカバリ可能なラッチは Oracle Rdb ロックと同じ機能を果たしますが、リカバリ可能なラッチの利点は Oracle Rdb ロックより少ない命令でロックをかけたり解除したりできることです。

リカバリ可能なラッチには、障害の発生したデータベースをデータベース・リカバリ（DBR）処理でリカバリするのに必要な情報が保持されます。

リカバリ可能なラッチを有効にするには構文は不要です。Oracle Rdb は、グローバルなバッファが有効なデータベースでは自動的にリカバリ可能なラッチを使用します。この機能によって、グローバルなバッファが有効なデータベースのパフォーマンスが改善されます。

3.8.8 読取り専用記憶領域

成長する見込みのない安定したデータを保持する記憶領域では、この記憶領域内のテーブルへのアクセスを読取り専用に変更できます。このためには、SQL の ALTER DATABASE 文を使用して記憶領域の読取り属性を読取り / 書込みから読取り専用に変更します。読取り専用に変更すると、ユーザーが読取り / 書込みトランザクションでテーブルの読取りに書込みのロックをかけたり、同様のタスクを実行する他のユーザーがページ・レベルまたは行レベルのロックを取得したりすることがなくなります。たとえば、DEPARTMENTS 記憶領域を読取り専用に変更するには、例 3-32（3-84）に示す文を使用します。

例 3-32 読取り / 書込みから読取り専用への状態変更

```
SQL> ALTER DATABASE FILENAME mf_personnel
  1> ALTER STORAGE AREA DEPARTMENTS READ ONLY;
```

読取り専用記憶領域は、データベース内の安定したデータを処理するための便利な方法です。読取り専用記憶領域内のテーブルに行を追加する場合は、そのときだけ SQL の ALTER DATABASE 文を使用しての読取り専用属性を読取り / 書込みに戻します。読取り専用記憶領域を含むデータベースのバックアップおよびリストア操作ストラテジの作成については、『Oracle Rdb7 Guide to Database Maintenance』を参照してください。

3.8.9 RDB$SYSTEM 読取り専用記憶領域の使用

デフォルトの RDB$SYSTEM 記憶領域内の安定したメタデータやテーブルを含むデータベースはサイズが変わらないので、デフォルトの RDB$SYSTEM 記憶領域の読取り / 書込み属性を読取り専用に変更できます。読取り専用に変更すると、読取り専用記憶領域のページと行のロックが解除されます。ただし、RDB$SYSTEM 記憶領域を読取り専用に設定すると、RMU Collect Optimizer_Statistics を使用しても論理領域のカーディナリティ以外のメタデータを変更できなくなります。データベースへのユーザー・アクセスの保護などのメタデータを変更するには、RDB$SYSTEM 記憶領域の読取り専用属性を読取り / 書込みに戻す必要があります。

注意： 記憶領域を読取り専用に設定すると、その領域内に作成したインデックスを含めて、領域内のオブジェクトは更新モードでは準備できなくなります。すなわち、読取り専用の領域に格納されたインデックスを更新モードで準備すると、操作は違反になります。たとえば、RDB$SYSTEM 記憶領域を読取り専用に設定すると、次のクエリーでは違反が発生します。

```
SQL> UPDATE EMPLOYEES
  1> SET MIDDLE_INITIAL = NULL
  2> WHERE EMPLOYEE_ID > '00200';
%RDMS-F-READONLY, data in a read-only storage area may not be
accessed for update
```

クエリーでは範囲検索が指定されているので、オプティマイザは EMPLOYEE_ID のソート・インデックスを使用して更新する行を検索します。このソート・インデックスは RDB$SYSTEM 領域内にあります。このクエリーを実行すると、Oracle Rdb はこのインデックス領域（RDB$SYSTEM）を更新モードで準備しようとするので、操作は違反になります。したがって、RDB$SYSTEM 記憶領域を読取り専用に設定する場合は、更新トランザクションに必要なインデックスを RDB$SYSTEM には置かないようにしてください。

RDB$SYSTEM 記憶領域を読取り専用に変更できます（例 3-33（3-85）を参照）。

例 3-33 RDB$SYSTEM 記憶領域の読取り / 書込みから読取り専用への状態変更

```
SQL> ALTER DATABASE FILENAME mf_personnel READ ONLY;
```

RDB$SYSTEM 記憶領域を読取り専用に設定すると、テーブルとインデックスのカーディナリティの自動更新は無効になります。この作業には、Qquery Optimizer に影響をおよぼすメタデータ・テーブルとインデックスのカーディナリティ（行の数）の手動による変更が抑制され、カーディナリティの更新に伴う I/O 操作がなくなるという副作用があります。

読取り専用の RDB$SYSTEM 記憶領域のすべてのテーブルとインデックスのカーディナリティを更新するには、RMU Collect Optimizer_Statistics コマンドを使用します。

注意： RMU Collect Optimizer_Statistics コマンドを使用するには、スナップショットを有効にし、許可する必要があります。カーディナリティ・テーブルのカウントは各テーブルの各行にアクセスして作成されるので、他の方法によるロックではデータベース全体がロックアウトされます。

たとえば、RDB$SYSTEM の記憶領域内を読取り専用に設定した後で新しい行が追加されたので、テーブルとインデックスの新しいカーディナリティを反映するように RDB$SYSTEM 記憶領域のテーブルとインデックスのカーディナリティを更新するとします。このためには、クエリー・オプティマイザがこの記憶領域内のメタデータに格納されたすべてのテーブルとインデックスに関する最新のカーディナリティの値に基づいて最適なクエリー・ストラテジを構成できるようにする必要があります。例 3-34（3-86）に示すようにメタデータを更新します。

例 3-34 RDB$SYSTEM 内のテーブルとインデックスのカーディナリティの更新

```
$ RMU/COLLECT OPTIMIZER_STATISTICS/STATISTICS=CARDINALITY mf_personnel
$ rmu -collect optimizer_statistics -statistics=cardinality mf_personnel
```

このコマンドは、システム・テーブル RDB$RELATIONS と RDB$INDICES の RDB$CARDINALITY 列のカーディナリティの値を更新します。

注意： 動的な最適化では、クエリーを実行する間に収集した統計に基づいて最適なクエリー・パフォーマンスを提供しようとします。インデックスとテーブルのカーディナリティが人為的に正しい値から大きく外されると、動的に収集した統計と矛盾し、その評価についてはまったく予測できないストラテジや順序が選択されることになります。したがって、カーディナリティを調整するとクエリー・オプティマイザのパフォーマンスは低下する確率が高くなります。

3.9 インデックス検索

インデックス検索は、Oracle Rdb が行へのアクセスに使用する 2 つの方法の 1 つです。もう 1 つの方法は順次アクセスです。

インデックス・アクセスには次の 3 種類があります。

- ダイレクト・インデックス・アクセス。Oracle Rdb は一意のインデックス・キーを使用してデータの位置を特定し、テーブルから直接アクセスします。この場合、キーは唯一の行に一致します。
- インデックスによる検索。Oracle Rdb は部分インデックス・キーまたは複製を使用してデータの位置を特定します。Oracle Rdb は、検索範囲の最下位のキー（下位 Ikey）に一致するすべての行と検索範囲の最上位のキー（上位 Ikey）に一致するすべての行を検

出するまでインデックスを使用してスキャンします。下位 Ikey と上位 Ikey の両方について、Oracle Rdb は dbkey を使用して実際の行を読み取ります。下位 Ikey と上位 Ikey の詳細は、付録 C（C-1）を参照してください。

- インデックスのみの検索。Oracle Rdb では、インデックスを使用して行エントリを検出します。この場合、データはインデックス・キーの一部なので、Oracle Rdb は dbkey を使用して行を読み取らず、インデックスからデータを返します。

I/O 操作の数によるコストの点では、インデックスのみの検索が最も効率的です。インデックスによる検索では、CPU のオーバーヘッドが最大になります。小規模なテーブルには順次アクセスが最適です。

インデックスは、データの検索だけでなく更新操作にも使用します。

更新トランザクションでは、インデックスが定義されていない他の列を更新する場合もあります。インデックス列を使用して行の位置を確認し、インデックスのない別の列を更新する場合は、インデックスの更新に必要なオーバーヘッドはありません。

インデックス列を更新するトランザクションは次のように動作します。

- 更新する行を参照するインデックス・ノードをロックします。
- トランザクションでテーブルに行を追加する場合は、分割の必要なインデックス・ノードをロックします。このトランザクションでは、インデックス自体も更新します。
- 読取り / 書込みトランザクションがこれらのインデックス・ノードを介してデータベースにアクセスしないようにします。

Oracle Rdb がインデックスを使用して dbkey を検出するのでなく、トランザクションで dbkey 自体を使用する場合は、インデックスを使用しないでプログラムから直接行にアクセスできます。dbkey の詳細は、『Oracle Rdb7 SQL Reference Manual』を参照してください。

3.9.1 インデックスのタイプ

Oracle Rdb では、次の 2 種類のインデックスを使用します。

- ソート・インデックス
  ソート・インデックスは、一般に最適な総合的アクセス方法です。完全一致、行の存在の決定、値の範囲検索、部分キーの検索で高パフォーマンスを示し、特にクラスタ化された場合（クラスタ B ツリー）のソート検索に優れています。動的なテーブルには最適です。
- ハッシュ・インデックス
  ハッシュ・インデックスは、完全一致で最高のパフォーマンスを提供し、特に非常に大規模なテーブルで使用した場合は優れています。このように大規模なテーブルでは B ツリー全体のサイズが大きいので、B ツリー・アクセスの方法はあまり適しません。

ソート・インデックス構造を定義すると、Oracle Rdb はインデックスを検索して dbkey を検出し、インデックス・キーから直接情報を取得してテーブル内の行にアクセスします。それ以外の場合は dbkey を使用してデータベース内の行を検出します。

ハッシュ・インデックス構造を定義すると、Oracle Rdb はハッシュ・バケットで検索キーと dbkey を検索し、行が存在するデータ・ページ番号に直接到達して、ダイレクト・アクセスによってテーブル内の行にアクセスします。ハッシュ・インデックスは、ランダムなダイレクト・アクセスによる検索キーの完全一致に最適です。

ソート・インデックス検索を使用する場合は、Adjustable Lock Granularity（ALG）が有効なことを確認してください。ALG を使用すると、他の検索や更新トランザクションがテーブル内のインデックス・ノードや行にアクセスする場合に、ロックの最小単位が最上位（論理領域、テーブル、またはインデックス全体）から始めてその中の次に小さなページのグループへと徐々にレベルを下げ、最下位（各行）に達するまでの間で自動的に調整されるので、データベースの並行性が大幅に向上します。しかし、他のトランザクションがソート・インデックスを介して該当する行にアクセスすると、更新され、ロックされたインデックス・ノードのロックにより、待機の状態が発生する恐れがあります。インデックス・ストラテジの詳細は、3.9.6 項（3-109）を参照してください。

データベース内の特定の列に対してソート・インデックスとハッシュ・インデックスを定義すると、オプティマイザは順次検索を使用した場合よりも少ないディスク・アクセスでデータを検索するためのアクセス方法を選択できます。データベース設計者は、対象となる列にいずれのインデックス・タイプが適しているかを慎重に見極める必要があります。主キーは常に適切なインデックスの候補ですが、一部の読取り専用のタスクでは他の列のインデックスも有効な場合があります。

ソート・インデックスの詳細は、3.9.6 項（3-109）を参照してください。ハッシュ・インデックスの詳細は、3.9.7 項（3-121）を参照してください。

3.9.2 インデックスの論理領域名

シングル・ファイル・データベースでは、Oracle Rdb はすべてのインデックス構造を .rdb ファイルの 1 つの論理領域に格納し、この論理領域に割り当てたページをデータ・ページとして扱います。すなわち、インデックスのデータ構造は他のデータ・ページと同じです。

マルチファイル・データベースでは、STORE 句を使用してテーブルのソート・インデックス構造を格納する記憶領域を割り当てます。Oracle Rdb は、テーブルの各インデックス構造を .rda ファイルの個々の論理領域に格納し、この論理領域に割り当てられたページをデータ・ページとして扱います。

論理領域名は RMU Analyze コマンドと RMU Dump コマンドに Larea=RDB$AIP 修飾子を指定した場合の出力に表示され、データベース構造を分析するために使用されます。

Oracle Rdb では、RDB$SYSTEM 記憶領域内に明示的または暗黙的に作成されたインデックスは同じテーブルに関する場合はすべて同じ論理領域を共有するという規則が適用されま

す。同じテーブルに関するすべてのインデックスで使用する論理領域の名前として、最初のインデックスの名前が割り当てられます。

例として、次に示す単純なテーブルとインデックスの定義について考察します。

```
SQL> CREATE TABLE TEST (I_COL integer, C_COL char);
SQL> CREATE INDEX IND_1 on T (I_COL);
SQL> CREATE INDEX IND_2 on T (C_COL);
```

インデックス IND_1 と IND_2 は、デフォルトで両方とも RDB$SYSTEM 記憶領域にマップされます。テーブル TEST のインデックス用に 1 つの論理領域が作成されており、最初に作成したインデックスの名前が付きますが（この場合は IND_1）、RDB$SYSTEM 記憶領域にマップされるテーブル TEST のすべてのインデックスの情報が格納されます（この場合は IND_1 と IND_2）。

その後インデックス IND_1 が削除され、別の記憶領域に再作成されても、データベース管理者は論理領域 IND_1 が RDB$SYSTEM（物理領域 1）内に引き続き存在していることがわかります。IND_2 インデックスのデータはそのまま存在するので、IND_1 という名前は残ります。したがって、論理領域 IND_1 はインデックス自体は RDB$SYSTEM 内に存在しなくても複数の記憶領域内に存在するので、混乱が起こる場合があります。

後でインデックス IND_2 も削除され、別の記憶領域に再作成されると、RDB$SYSTEM 内のオリジナルの IND_1 論理領域は使用されないので削除されます。

インデックスが RDB$SYSTEM 以外の記憶領域にマップされる場合、インデックスは常に別々のプライベート論理領域に割り当てられます。

注意：オラクル社では、新しいデータベースをマルチファイル・データベースとして作成し、すべてのインデックスとテーブルにストレージ・マップを置くことをお薦めします。これで、このような混乱の問題が回避され、データベースの物理的な再構築にも柔軟に対応できます。

上記のインデックスに関する論理領域共有の規則は、シングルファイル・データベース・モデルの要素であり、古いバージョンの Oracle Rdb の上位互換性を守るために保持されます。この規則の利点は、シングルファイル・データベースと常に RDB$SYSTEM 記憶領域に格納されるシステム・テーブルの領域を節約することです。

3.9.3 インデックスの圧縮

インデックスを圧縮するように指定できます。**圧縮型インデックス**では、インデックス・ノード内の情報のサイズが小さくなるので、データに必要な領域が小さくなります。インデックスの圧縮には、次の 4 種類があります。

- 接頭辞と接尾辞の圧縮（3.9.3.1 項（3-90）を参照）

- SIZE IS によるセグメントの切捨て（3.9.3.2 項（3-90）を参照）
- MAPPING VALUES による圧縮（3.9.3.3 項（3-91）を参照）
- run-length による圧縮（3.9.3.4 項（3-92）を参照）

インデックスの圧縮には、次の利点があります。

- 一部のアプリケーションではストレージ要求が大幅に縮小されます。
- 多くのユーザー・インデックス・ノードがバッファに入るので、データを検索するための I/O 操作が減少します。
- 理論上多くのデータが 1 つのインデックスに含まれるので、インデックスのみの検索の効率が向上します。

3.9.3.1 接頭辞と接尾辞の圧縮

接頭辞と接尾辞の圧縮は、ソート・インデックスの場合にのみ自動的に発生します。データ行を指す最低レベルのインデックス（レベル 1 のインデックス・ノード）では、接頭辞の圧縮のみが発生します。

これより上のレベルのインデックスでは、接頭辞と接尾辞の圧縮が発生します。このように上位レベルのインデックス・ノードでは、接頭辞の圧縮によって連続するインデックス・キー間でインデックス・キーの先頭の上位バイトが等しい場合はこれが削除されます。接尾辞の圧縮では、インデックス・キーの末尾から無意味なバイトを削除します。インデックス・キーの接頭辞と接尾辞の詳細は、3.9.5.1 項（3-97）の例 3-46（3-106）の説明を参照してください。

3.9.3.2 SIZE IS によるセグメントの切捨て

CREATE INDEX 文の SIZE IS 句を使用して、特定のキーの先頭の n 文字をインデックスに使用するように指定します。たとえば、通常は先頭の 20 文字が一意の 100 バイトの列をインデックスに使用する場合は、先頭の 20 文字だけをインデックスに使用するように指定すると、エントリあたり 80 バイト分を節約できます。セグメントの切捨ては、ソート・インデックスにのみ指定できます。

データ型が CHAR または VARCHAR の列に対して SIZE IS で圧縮したインデックスを作成するには、インデックスに使用する列に対して SQL の CREATE INDEX 文の SIZE IS 句を使用します（例 3-35（3-90）を参照）。

例 3-35 CHAR データ型の列の SIZE IS によるインデックス圧縮の設定

```
SQL> CREATE INDEX EMP_LAST_NAME ON EMPLOYEES
  1>  (LAST_NAME SIZE IS 10)
  2>   TYPE IS SORTED;
```

SIZE IS による圧縮型インデックスを作成する場合は、UNIQUE 句を指定できます。しかし、SIZE IS による圧縮型インデックスで UNIQUE 句を指定すると、インデックス・キー値の切捨てによってキーの値が一意でなくなる恐れがあります。この場合は、インデックスの定義、あるいは挿入や更新の文がエラーになります。たとえば、EMP_LAST_NAME インデックスを一意として定義し、"Kilpatrick" という姓の従業員と "Kilpatricks" という姓の従業員の行を挿入しようとすると、SQL は次のエラーを返します。

```
%RDB-E-NO_DUP, index field value already exists; duplicates not allowed for
EMP_LAST_NAME
```

3.9.3.3 MAPPING VALUES による圧縮

MAPPING VALUES による圧縮を指定すると、Oracle Rdb は列の値をよりコンパクトなエンコード形式に変換することで、すべての数値列のインデックスを格納するために必要なビット数を削減します。この種の圧縮は、SQL の CREATE INDEX 文の MAPPING VALUES 句で指定します。MAPPING VALUES による圧縮はテキスト列には指定できないのでご注意ください。MAPPING VALUES による圧縮は、ソート・インデックスとハッシュ・インデックスに指定できます。

MAPPING VALUES 句では、列に格納できる値の範囲を指定する必要があります。したがって、MAPPING VALUES 句は指定した範囲外の値を許可しないので、制約として機能します。

データ型が TINYINT、SMALLINT、INTEGER の列に対して MAPPING VALUES で圧縮したインデックスを作成するには、インデックスに使用する列に対して SQL の CREATE INDEX 文の MAPPING VALUES 句を使用します（例 3-36（3-91）を参照）。整数の圧縮型インデックスには、UNIQUE 句を指定できます。例 3-36（3-91）で、PRODUCT_ID、YEAR_NUMBER、PRODUCT_DESCR は UNIQUE 句で定義された 3 つの列です。

例 3-36 SMALLINT データ型の列の MAPPING VALUES によるインデックス圧縮の設定

```
SQL> CREATE UNIQUE INDEX PS_DATE_2 ON PRODUCT_SCHEDULE
  1> (PRODUCT_ID,
  2>  YEAR_NUMBER  MAPPING VALUES 1970 to 2070,
  3>  PRODUCT_DESCR SIZE IS 20);
```

すべてのインデックス・キーには、Oracle Rdb によって 1 ビットが追加されます。多くのインデックス・キー値では、このビットを含む 1 バイトが先頭に追加されます。このバイトは最下位ビット以外は意味がありません。MAPPING VALUES と ENABLE COMPRESSION を指定した数値のインデックス・キーでは、この追加のビットが最上位ビットに入るので、余分なバイトは必要ありません。多くのインデックス・キー値では、この追加のビットまたはバイトは、インデックス・キーの実際のデータ値がある場合はクリアされ、データ値がない場合や NULL の場合はセットされます。

NULL ビットは、ソート・インデックスでは NULL インデックス・キーが非 NULL インデックス・キーの後になることを保証するために使用され、ハッシュ・インデックスではキーが NULL の場合にハッシュ・アルゴリズムのデータを提供するために使用されます。

Oracle Rdb は、すべてのインデックス・キーに対して自動的に NULL ビット圧縮を使用します。

3.9.3.4 run-length による圧縮

run-length による圧縮を指定すると、Oracle Rdb はテキスト・データ型の空白文字（オクテット）と非テキスト・データ型のバイナリのゼロを圧縮します（様々なキャラクタ・セットには様々な空白文字の表現があります。Oracle Rdb は、インデックス値を構成する列のキャラクタ・セットの空白文字の表現を圧縮します）。run-length による圧縮は、空白文字やバイナリのゼロが多い場合は非常に有効です。run-length による圧縮は、SQL の CREATE INDEX 文の ENABLE COMPRESSION MINIMUM RUN LENGTH 句で指定します。run-length による圧縮は、ハッシュ・インデックスとソート・インデックスに指定できます。run-length による圧縮は、3.9.4 項（3-97）で説明するようにシステム・インデックスにも指定できます。

データ型が CHAR または VARCHAR の列に対して run-length による圧縮型インデックスを作成するには、SQL の CREATE INDEX 文の ENABLE COMPRESSION MINIMUM RUN LENGTH 句を指定します（例 3-37（3-92）を参照）。MINIMUM RUN LENGTH 句に指定する値は、Oracle Rdb が圧縮する最小の列の長さを示します。たとえば、MINIMUM RUN LENGTH 2 を指定すると、Oracle Rdb は 2 つ以上の空白の連続または 2 つ以上のバイナリ・ゼロの連続を圧縮します。

例 3-37 CHAR データ型の列の MINIMUM RUN LENGTH によるインデックス圧縮の設定

```
SQL> CREATE TABLE TELEPHONE_LIST
  1>  (NAME CHAR (60),
  2>   ADDRESS CHAR (141),
  3>   TELEPHONE_NUMBER CHAR (12),
  4>   TIME_DATE_OF_CALL DATE);
SQL>
SQL> CREATE INDEX TELEPHONE_CUSTOMER ON TELEPHONE_LIST
  1>  (NAME, ADDRESS, TELEPHONE_NUMBER)
  2>  ENABLE COMPRESSION
  3>  (MINIMUM RUN LENGTH 2);
```

Oracle Rdb が空白またはバイナリ・ゼロの連続を圧縮すると、連続する空白またはゼロは、MINIMUM RUN LENGTH で指定した空白またはバイナリ・ゼロの数と、この連続で圧縮した空白またはバイナリ・ゼロの数の情報を含む別の 1 バイトに置き換わります。

run-length による圧縮では、ストレージをかなり節約できる場合があります。例 3-37（3-92）で作成した TELEPHONE_CUSTOMER インデックスについて考察します。TELEPHONE_CUSTOMER インデックスが run-length による圧縮なしで定義された場合、

各非圧縮型インデックス・キーの長さは 216 バイトです。個々の 216 バイトの非圧縮型インデックス・キーの構成は次のとおりです。

- 60 バイトの NAME 列（インデックスの最初のセグメント）
- 141 バイトの ADDRESS 列（インデックスの 2 番目のセグメント）
- 12 バイトの TELEPHONE_NUMBER 列（インデックスの 3 番目のセグメント）
- 他の 3 バイト（3 つの各インデックス・キー・セグメントに、インデックス・キー・セグメントの NULL ビットを含む 1 バイトが付加されます）

NAME 列に格納される名前の長さが平均で 13 文字しかない場合は、run-length による圧縮を使用するとインデックス・キーを格納するスペースが大幅に削減されます。

例 3-38（3-93）に、TELEPHONE_CUSTOMER インデックスに対して MINIMUM RUN LENGTH2 が指定された場合に Oracle Rdb がインデックス・エントリ "Terry L Smith" を圧縮する方法を示します。例 3-38（3-93）では、番号記号（#）は空白を表し、c は圧縮情報を含むバイトを表します。非圧縮型インデックス・キー・セグメントの長さは 60 文字ですが、13 文字の名前が格納されており、残りの 47 文字は空白です。MINIMUM RUN LENGTH 2 が指定された場合、Oracle Rdb は末尾の 47 文字の空白を 2 つの空白（MINIMUM RUN LENGTH 句で指定した値）で置き換え、圧縮情報を含む 1 バイトを追加します。圧縮型インデックス・キー・セグメントのサイズはわずかに 16 バイトです。これは、非圧縮型インデックス・キー・セグメントに対して 44 バイトの節約になります。

例 3-38 TELEPHONE_CUSTOMER インデックスの NAME 列に関するインデックス・キーの圧縮

```
Uncompressed index key entry:
Terry#L#Smith#################################################
Compressed index key entry with MINIMUM RUN LENGTH 2:
Terry#L#Smith##c
^           ^  ^                                             ^
|           |  |                                             |
|           |  |                                             |
1          13 16                                            60
Number of characters in the index key
```

同様に、ADDRESS 列に格納された住所の長さが平均で 18 文字しかない場合は、run-length による圧縮を使用するとインデックス・キーを格納する領域が大幅に削減されます。ADDRESS 列の非圧縮型インデックス・キー・セグメントの長さが 18 文字の場合、末尾の 123 文字は空白です（ADDRESS 列のサイズ 141 文字を基準として）。

MINIMUM RUN LENGTH 2 が指定された場合、Oracle Rdb は末尾の 123 文字の空白を 2 つの空白（MINIMUM RUN LENGTH 句で指定した値）で置き換え、圧縮情報を含む 1 バイトを追加します。圧縮型インデックス・キー・セグメントのサイズはわずかに 21 バイトです（18 バイトのインデックス・キー・セグメントと MINIMUM RUN LENGTH 2 句によ

る 3 バイト）。これは、非圧縮型インデックス・キー・セグメントに比べて 120 バイトの節約になります。

TELEPHONE_CUSTOMER インデックスの個々の圧縮型インデックス・キーは、平均サイズが 52 バイトで構成は次のとおりです。

- 16 バイトの NAME 列（インデックスの最初のセグメント）
- 21 バイトの ADDRESS 列（インデックスの 2 番目のセグメント）
- 12 バイトの TELEPHONE_NUMBER 列（インデックスの 3 番目のセグメント）
- 他の 3 バイト（3 つのインデックス・キー・セグメントごとにそのインデックス・キー・セグメントの NULL ビットを含む 1 バイトが付加されます）

TELEPHONE_CUSTOMER インデックスに 100,000 エントリを格納するように計画するとします。非圧縮形式でインデックス・エントリを格納した場合、インデックス・エントリの記憶領域は次のように 42188 ディスク・ブロックになります。

```
100,000 index entries × 216 bytes per entry = 21,600,000 bytes
21,600,000 bytes / 512 bytes per block = 42187.5 blocks
```

しかし、インデックス作成時に MINIMUM RUN LENGTH 2 を指定した場合は、100,000 インデックス・キー・エントリの記憶領域は次のように約 10157 ブロックです（一部のインデックス・キー・エントリは平均の 52 バイトのエントリより大きい場合や小さい場合があります）。

```
100,000 index entries × 52 bytes per entry = 5,200,000 bytes
5,200,000 bytes / 512 bytes per block = 10156.3 blocks
```

この場合、run-length による圧縮で節約されたストレージは、約 32,030 ブロックです（42188 ブロック -10157 ブロック）。表 3-11（3-94）に、TELEPHONE_CUSTOMER インデックスに 10 万または 1000 万のインデックス・キー・エントリがある場合の RUN LENGTH による圧縮で節約される記憶領域を示します。

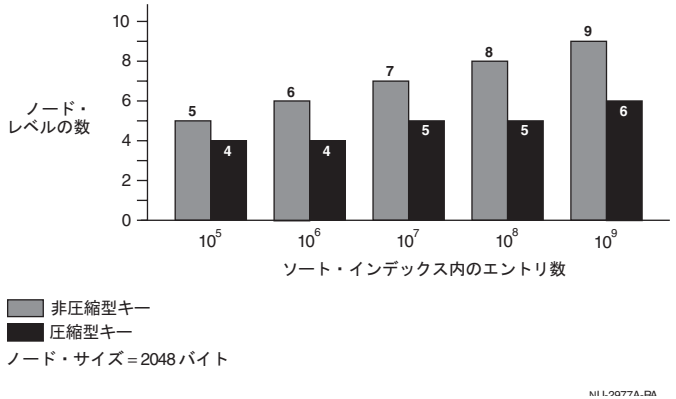
表 3-11 TELEPHONE_CUSTOMER インデックスの RUN LENGTH による圧縮で節約される記憶領域

インデックス・エントリの数	非圧縮型インデックス・エントリ	圧縮型インデックス・エントリ	節約
100,000	42.188 ブロック	10,157 ブロック	32,030 ブロック
10,000,000	4,218,750 ブロック	1,015,625 ブロック	3,203,125 ブロック

表 3-11（3-94）から、非圧縮型キーのインデックスは圧縮型キーのインデックスより大きな領域が必要なことがわかります。インデックス・キーの圧縮によるもう 1 つの利点は、個々のソート・インデックス・ノードにより多くのインデックス・キーを格納できることです。

図 3-5（3-95）に、ノード・サイズ 2048 バイトで作成したソート・インデックスを示します。100,000 の非圧縮型インデックス・キー・エントリをソートする場合、インデックス・ノードのレベルは 5 です。しかし、100,000 の圧縮型インデックス・キー・エントリをソートする場合、インデックス・ノードのレベルは 4 です。インデックス・ノードのレベルの値が低くなるとインデックス・キー・エントリにアクセスするために必要な I/O 操作の数も減るので、I/O の観点からも効果があります。

図 3-5 RUN LENGTH による圧縮を指定した場合のインデックス・ノードのレベル低下



各ノードに多くのインデックス・キーを格納する影響の 1 つとして、並行性低下の可能性があります。多くのインデックス・キーが 1 つのノードに入ると、行を更新するたびに多くのインデックスの値をロックすることになります。インデックス・キーのサイズを縮小すると並行性が低下する場合は、ノードのサイズを縮小して各ノードのインデックス・キーの数を減らすと、並行性を向上できます。

インデックスに格納するデータについて十分理解した上で、インデックスの MINIMUM RUN LENGTH の値を指定してください。不用意に MINIMUM RUN LENGTH の最適値より小さな値を指定すると、圧縮型インデックスが非圧縮型インデックスより大きくなる場合もあります。次の例では、インデックスに対して MINIMUM RUN LENGTH 1 が指定されています。番号記号（#）はバイナリ・ゼロを表し、c は圧縮情報を含むバイトを表します。

```
Uncompressed index key entries:
1#2#3#4#5#6
2#3#4#5#6#7
3#4#5#6#7#8
Compressed index key entries with MINIMUM RUN LENGTH 1:
1#c2#c3#c4#c5#c6
2#c3#c4#c5#c6#c7
3#c4#c5#c6#c7#c8
^          ^    ^
|          |    |
|          |    |
1         11   16
Number of characters in the index key
```

この特定のインデックスでは、MINIMUM RUN LENGTH 1 が指定された場合に行われる圧縮によって、圧縮型インデックス・キーが非圧縮型インデックス・キーより大きくなります（圧縮型キーが 16 バイトで非圧縮型キーが 11 バイト）。MINIMUM RUN LENGTH 1 を指定すると Oracle Rdb は各バイナリ・ゼロを MINIMUM RUN LENGTH の値で圧縮するので、圧縮型インデックス・キーの方が大きくなります。この結果、Oracle Rdb は各バイナリ・ゼロを 1 つのバイナリ・ゼロと圧縮したバイナリ・ゼロの情報を含む追加の 1 バイトに置き換えます（1 つのバイナリ・ゼロが 2 文字に置き換わる）。非圧縮型インデックス・キー・エントリの記憶領域が 11 バイトであるにもかかわらず、Oracle Rdb は 16 バイトの圧縮型インデックス・キー・エントリを格納する可能性があることに注意してください。Oracle Rdb では、255 バイト未満の圧縮型または非圧縮型のインデックス・キー・エントリを格納できます。圧縮型インデックス・キーのサイズが 255 バイトを超えた場合は、次のエラー・メッセージが表示されます。

```
%RDMS-F-IKEYOVFLW, compressed IKEY for index index-name exceeds 255 bytes
```

このメッセージは、インデックス・キーがインデックスに格納できないことを表します。この場合は、インデックスを削除して再定義してください。新しいインデックスで同じエラー・メッセージが表示されないようにするには、新しいインデックスで run-length による圧縮を無効にするか、run-length による圧縮を有効にして minimum run-length の値を大きくする必要があります。

run-length による圧縮を指定する場合は、Oracle Rdb が圧縮する文字列の最小の長さは指定できますが、どの文字を圧縮するかは指定できないことに注意してください。どの文字を圧縮するかは、Oracle Rdb が決定します。

SQL の SHOW INDEXES 文によって、定義済みのインデックスの圧縮に関する特性が表示されます。RMU Analyze Indexes コマンドにデフォルトの Option=Normal 修飾子を指定すると、各インデックス・ノードがソート・インデックスに使用する領域のサイズが表示されます。3.9.5.1 項（3-97）を参照してください。ノードごとに表示される値をインデックス圧縮の前と後で比較すれば、インデックス圧縮の効果を確認できます。ソート・インデックスは事前に割り当てられた構造なので、この情報を表示できます。動的なハッシュ・インデックス構造では RMU Analyze Indexes Option=Normal による表示の Used/Avail の値が同じなの

で、圧縮情報は表示できません。ハッシュ・インデックス構造では、圧縮の前と後にコマンド行で Option=Debug 修飾子を使用して個々の値を調べて差を確認する必要があります。

圧縮型インデックスの詳細は、『Oracle Rdb7 Guide to Database Design and Definition』と『Oracle Rdb7 SQL Reference Manual』を参照してください。

3.9.4 システム・インデックスの圧縮

データベースを作成する場合は、Oracle Rdb でシステム・インデックスを圧縮するように指定できます。このためには、SQL の CREATE DATABASE 文の SYSTEM INDEX COMPRESSION IS ENABLED 句を使用します。既存のデータベースのシステム・インデックスの圧縮を有効にするには、データベースをエクスポートしてからインポートし、IMPORT 文で SYSTEM INDEX COMPRESSION IS ENABLED 句を指定します。

システム・インデックスの場合は、Oracle Rdb は run-length による圧縮を使用してテキスト・データ型の空白と非テキスト・データ型のバイナリ・ゼロ（ASCII NUL とも呼ばれる）の連続を圧縮します。run-length による圧縮では、テキスト・データ型の空白または非テキスト・データ型のバイナリ・ゼロが 2 つ以上続く場合に圧縮を行います。run-length による圧縮の詳細は、3.9.3.4 項（3-92）を参照してください。

システム・インデックスを圧縮すると、ストレージが削減され、I/O が改善されます。複数のアプリケーションが同時にデータ定義を実行する頻度が高くない限り、システム・インデックスを圧縮してください。

3.9.5 インデックス情報の収集

この項では、RMU Analyze コマンドと RMU Show コマンドについて説明します。これらのコマンドを使用すると、ソート・インデックスとハッシュ・インデックスに関する情報を収集できます。RMU Analyze コマンドの使用方法の概要は、2.1 項（2-2）を参照してください。インデックス分析に関するアドバイスは、8.1.3.2 項（8-25）を参照してください。

3.9.5.1 RMU Analyze Indexes による表示

この項では、RMU Analyze コマンドで Indexes 修飾子と Option [= Normal、Full、Debug のいずれか] 修飾子を指定した場合の出力の書式と内容について説明します。

RMU Analyze Indexes Option=Normal コマンドの使用

RMU Analyze Indexes コマンドで Option=Normal 修飾子を使用し、EMPLOYEES_HASH ハッシュ・インデックスを指定すると、Oracle RMU では例 3-39（3-98）に示す情報が表示されます。

例 3-39 RMU Analyze Indexes Option=Normal コマンド（ハッシュ・インデックスの場合）

```
$ RMU/ANALYZE/INDEXES mf_personnel EMPLOYEES_HASH /OPTION=NORMAL
-------------------------------------------------------------------------------
 Indices for database  - $DUA0:[ORION]MF_PERSONNEL.RDB;
-------------------------------------------------------------------------------
 Hashed index EMPLOYEES_HASH for relation EMPLOYEES duplicates not allowed
 Max Level: 1, (1)
Nodes: 69, (2)
Used/Avail: 2797/2797 (100%), (3)
Keys: 100, (4)
Records: 100 (5)
-------------------------------------------------------------------------------
```

次のリストでは、RMU Analyze Indexes Option=Normal の表示について説明します。番号は例 3-39（3-98）のフィールドに対応しています。

(1) Max Level:
インデックス・レベルの最大数（1）

(2) Nodes:
インデックス内のノードの総数（69）。ここではハッシュ・インデックスに関する表示なので、ノードはハッシュ・バケットです。

(3) Used/Avail:
インデックスで使用するバイト数（2797）、使用可能なバイト数（2797）、インデックスによる領域使用率（100%）
ハッシュ・インデックスでは、Used/Avail の値にはインデックスのヘッダーとトレーラの情報が入ります。ハッシュ・インデックスは動的な構造なので、Used/Avail の値は常に同じであり、使用率は常に 100% です。
ソート・インデックスでは、Used/Avail の値にインデックスのヘッダーとトレーラの情報は入りません。この情報は 32 バイトを使用します。

(4) Keys:
インデックス内の一意キーの総数（100）。この例では重複が許可されないので、100 のデータ行には 100 の一意キーがあります。

(5) Records:
インデックス内で一意キーを持つインデックス・レコードの総数（100）

RMU Analyze Indexes コマンドで Option=Normal 修飾子と DEPARTMENTS_INDEX ソート・インデックスを指定すると、Oracle RMU では例 3-40（3-99）に示す情報が表示されます。

例 3-40 RMU Analyze Indexes Option=Normal コマンド（ソート・インデックスの場合）

```
$ RMU/ANALYZE/INDEXES mf_personnel DEPARTMENTS_INDEX /OPTION=NORMAL
-------------------------------------------------------------------------------
 Indices for database  - $DUA0:[ORION]MF_PERSONNEL.RDB;
-------------------------------------------------------------------------------
 Index DEPARTMENTS_INDEX for relation DEPARTMENTS duplicates not allowed
 Max Level: 1, Nodes: 1, Used/Avail: 188/398 (47%), Keys: 26, Records: 26
-------------------------------------------------------------------------------
```

DEPARTMENTS_INDEX ソート・インデックスに関する例 3-40（3-99）では、最大レベルが 1 で、1 つのノード（ルート・ノード）が存在し、インデックスは使用可能な 398 バイトのうち 188 バイトを使用し、ノードの使用率は 47% です。重複が許可されないので、26 のデータ行には 26 の一意キーがあります。ソート・インデックスでは、割り当てられた領域のサイズはデフォルトで 430 バイトですが、通常は実際にキーの格納にはこれほど大きな領域を使用しません。したがって、通常は領域使用率は 100% 未満です。

ソート・インデックスでは、Used/Avail の値にインデックスのヘッダーとトレーラの情報は入りません。これは 32 バイトに相当します。したがって、このソート・インデックスに事前に割り当てられた実際のノード・サイズはソート 430 バイトであり 398 バイトではありません。ソート・インデックスはハッシュ・インデックスとは違って事前に割り当てられた構造なので、この値は実際のパーセンテージの値です。

RMU Analyze Indexes コマンドで Option=Normal 修飾子と重複を許可するランク付きのソート・インデックスを指定すると、Oracle RMU では例 3-41（3-99）に示す情報が表示されます。

例 3-41 RMU Analyze Indexes Option=Normal コマンド（ランク付きソート・インデックスの場合）

```
$ RMU/ANALYZE/INDEXES mf_personnel DEGREES_YEAR_RANKED /OPTION=NORMAL
-------------------------------------------------------------------------------
 Indices for database  - $DUA0:[ORION]MF_PERSONNEL.RDB;
-------------------------------------------------------------------------------
 Index DEGREES_YEAR_RANKED for relation DEGREES duplicates allowed
 Max Level: 2, Nodes: 5, Used/Avail: 772/1990 (39%), Keys: 25, Records: 3
     Duplicate nodes: 0, Used/Avail: 0/0 (0%), Keys: 18, Maps: 18, Records: 162
-------------------------------------------------------------------------------
```

インデックスで重複を許可すると、RMU Analyze Indexes コマンドによって次の追加情報が表示されます。

- Duplicate nodes
  ランク付きソート・インデックスでは、オーバーフロー・ノードの数。この数は、インデックスに重複があってもゼロ（0）の場合があります。ランクなしソート・インデックスでは、重複ノードの数。ハッシュ・インデックスでは、重複ノードの数。

- Used/Avail
  重複ノードで使用するバイト数と重複ノードで使用可能なバイト数（インデックスの重複ノード内で使用された領域のパーセンテージ）。ランク付きソート・インデックスでは、重複ノードの数がゼロの場合はこの値がゼロ（0）になることがあります。
- Keys
  インデックス内の重複キーの総数。
- Maps
  ランク付きソート・インデックスのみで、記憶領域内でソートされた重複インデックス・キー・データを指す dbkey を表すビット・マップの数。ランクなしソート・インデックスとハッシュ・インデックスでは、このフィールドが表示されません。
- Records
  インデックス内の重複レコードの総数。

RMU Analyze Indexes コマンドで修飾子 Option=Normal を使用し、重複を許可する JOB_HISTORY_HASH ハッシュ・インデックスを指定すると、Oracle RMU では例 3-42（3-100）に示す情報が表示されます。

例 3-42 RMU Analyze Indexes Option=Normal コマンド（ハッシュ・インデックスで重複を許可する場合）

```
$ RMU/ANALYZE/INDEXES mf_personnel JOB_HISTORY_HASH /OPTION=NORMAL
-------------------------------------------------------------------------------
 Indices for database  - $DUA0:[ORION]MF_PERSONNEL.RDB;
-------------------------------------------------------------------------------
 Hashed index JOB_HISTORY_HASH for relation JOB_HISTORY duplicates allowed
 Max Level: 1, Nodes: 69, Used/Avail: 2797/2797 (100%), Keys: 100, Records: 20
     Duplicate nodes: 80, Used/Avail: 2992/7360 (41%), Keys: 80, Records: 254
-------------------------------------------------------------------------------
```

JOB_HISTORY_HASH ハッシュ・インデックスに関する例 3-42（3-100）では、最大レベルが 1 で、69 のノード（ハッシュ・バケット）が存在し、2797 バイトすなわち使用可能な領域の 100% を使用しています。インデックス内には合計 100 個のキーがあります。一意キーを持つ 20 のレコードおよび一意でないキーを持つ 254 のレコードがあります。100 個のキーのうち 80 個は一意ではありません。非一意キーあたりの平均重複数は 2.175 すなわち (254/80) -1、インデックス全体で計算した場合の重複数は 1.74 すなわち ((254+20)/100) -1 です。さらに、80 の重複ノード・レコードは使用可能な 7360 バイトのうち 2992 バイトを使用しており、使用率は 41% です。

ソート・インデックスのレベル数の値は、B ツリー・インデックスの深さのレベル数を示します。

ハッシュ・インデックスのレベル数の値は、インデックスのハッシュ・バケット・オーバーフロー・チェーンの最大長を示します。ハッシュ・バケット内に新しいエントリを作成する十分な領域がページ上にない場合はオーバーフローが発生し、隣接するページに十分な領域があればそこにオーバーフロー・ハッシュ・バケットが作成されます。親レコードと子レ

コードの両方のハッシュ構造全体が PLACEMENT VIA INDEX 句を使用するデータ行と同じページ上にある場合は、オーバーフローが一般化する恐れがあります。インデックス・レベル数の値は、システムがインデックスを使用して検索を行う場合に、特定の値の検出に要する I/O 操作の回数を示す場合とそうでない場合があります。しかし、RMU Analyze Placement コマンドで Option=Full または Option=Debug 修飾子を使用した結果を参照し、最小 I/O パス長と頻度のヒストグラムか最小 I/O パス長の値を調べれば、必要な情報がすべてバッファ内に存在するかどうかを正確に推定できます。

Oracle Rdb がソート・インデックスの最下位レベルとデータ・ページを読み取る場合は、検索には 3 回の I/O 操作が必要です。しかし、インデックスの最上位レベルは通常はバッファ内にあります。十分なバッファがある場合は、下位レベルもバッファに残ります。この場合、行の検索に必要な I/O 操作は 1 回だけです。

SQL の CREATE STORAGE MAP 文と ALTER STORAGE MAP 文の PLACEMENT VIA INDEX 句を使用してデータと同じ記憶領域内のデータと同じページに格納されたハッシュ・インデックスでは、データ行を検索するのに最低 1 回の I/O 操作が必要です。重複が許可される場合は、重複ノード・レコードが作成されます。多くの重複レコードがあり、ページ・サイズが小さすぎると、データ行はハッシュされたページに配置されますが、ハッシュ構造（ハッシュ・バケット）の一部は隣接するページにオーバーフローします。ハッシュ・バケットにエントリを追加する十分な領域がないためです。時間の経過とともに、データベース・ページに空きがなくなるので、オーバーフローが増大する可能性があります。結果として、必要な情報を収集するのにより多くの I/O 操作が必要になるので、パフォーマンスが低下する恐れがあります。

RMU Analyze Indexes Option=Normal コマンドによる表示には、圧縮型インデックスの詳細を示す 1 行が追加されます（例 3-43（3-101）を参照）。表示の最後の行は、インデックスが圧縮型インデックスの場合にのみ表示されます。

例 3-43 RMU Analyze Indexes Option=Normal コマンド（圧縮型インデックスの場合）

```
$ RMU/ANALYZE/INDEXES test_db COMPRESSED_IND /OPTION=NORMAL
0------------------------------------------------------------------------------
 Index COMPRESSED_IND for relation NEW_TABLE duplicates allowed
 Max Level: 1, Nodes: 1, Used/Avail: 155/398 (39%), Keys: 8, Records: 8
     Duplicate nodes: 0, Used/Avail: 0/0 (0%), Keys: 0, Records: 0
     Total Comp/Uncomp IKEY Size: 179/256, Compression Ratio:  .70
0
0------------------------------------------------------------------------------
```

例 3-43（3-101）の最後の行で、179 は圧縮型リーフ・インデックス・キーの総バイト数を示します（レベル 1 のノードのみが存在します）。256 は、圧縮されていない場合に消費する総バイト数を示します。圧縮ファクタが 1.0 を超える場合は、圧縮型インデックス・キーが非圧縮型インデックス・キーよりも大きな領域を占めることになります。3.9.3.4 項（3-92）に、圧縮を使用した場合に、圧縮型インデックス・キーがどのようにして非圧縮型インデックス・キーより大きな領域を占めるかの説明があります。

RMU Analyze Indexes Option=Full コマンドの使用

RMU Analyze Indexes コマンドで Option=Full 修飾子を指定すると、全レベルのインデックスに関する情報が表示されます。DEPARTMENTS_INDEX ソート・インデックスでは、インデックスのレベルは唯一です（例 3-44（3-102）を参照）。

例 3-44 RMU Analyze Indexes Option=Full コマンドの使用（ソート・インデックスの場合）

```
$ RMU/ANALYZE/INDEXES mf_personnel DEPARTMENTS_INDEX /OPTION=FULL
-------------------------------------------------------------------------------
 Indices for database  - $DUA0:[ORION]MF_PERSONNEL.RDB;
-------------------------------------------------------------------------------
 Index DEPARTMENTS_INDEX for relation DEPARTMENTS duplicates not allowed
 Max Level: 1, Nodes: 1, Used/Avail: 188/398 (47%), Keys: 26, Records: 26
-------------------------------------------------------------------------------
```

インデックスのレベルが複数の場合は、レベルごとの情報が表示されます。このインデックスは 1 レベルだけなので、情報が繰り返されます。

RMU Analyze Indexes Option=Debug コマンドの使用

RMU Analyze Indexes コマンドで Option=Debug 修飾子を指定すると、個々のインデックス・ノードとインデックス・レコードに関する詳細情報が表示されます。例 3-45（3-102）に、EMPLOYEES_HASH ハッシュ・インデックスに関する出力を示します。

例 3-45 RMU Analyze Indexes Option=Debug コマンド（ハッシュ・インデックスの場合）

```
$ RMU/ANALYZE/INDEXES mf_personnel EMPLOYEES_HASH /OPTION=DEBUG
0-------------------------------------------------------------------------------
0
0 Indices for database  - $DUA0:[ORION]MF_PERSONNEL.RDB;
0
0   ID   Hash Dup Name                            Relation
0-------------------------------------------------------------------------------
4      62   T   F  EMPLOYEES_HASH                 EMPLOYEES
4      60   T   F  EMPLOYEES_HASH                 EMPLOYEES
4      58   T   F  EMPLOYEES_HASH                 EMPLOYEES
0
0-------------------------------------------------------------------------------
0
0   ID        DB KEY       LEVEL    SIZE    KEY-LEN    KEY
0-------------------------------------------------------------------------------
0
5  58 0058:0000000002:002     1 0051/0051
5  58 0063:0000000002:001     0    0/   0     (006)"003030313635"
5  58 0063:0000000002:003     0    0/   0     (006)"003030313930"
5  58 0058:0000000005:002     1 0032/0032
5  58 0063:0000000005:001     0    0/   0     (006)"003030313837"
5  58 0058:0000000007:002     1 0070/0070
```

```
5  58 0063:0000000007:001     0    0/   0     (006)"003030313639"
5  58 0063:0000000007:003     0    0/   0     (006)"003030313736"
5  58 0063:0000000007:004     0    0/   0     (006)"003030313938"
5  58 0058:0000000011:002     1 0032/0032
.
.
.
5  60 0060:0000000003:002     1 0032/0032
5  60 0064:0000000003:001     0    0/   0     (006)"003030323133"
5  60 0060:0000000004:002     1 0032/0032
5  60 0064:0000000004:001     0    0/   0     (006)"003030323139"
5  60 0060:0000000005:002     1 0051/0051
5  60 0064:0000000005:001     0    0/   0     (006)"003030323235"
5  60 0064:0000000005:003     0    0/   0     (006)"003030323430"
5  60 0060:0000000006:002     1 0032/0032
.
.
.
5  62 0062:0000000008:002     1 0032/0032
5  62 0065:0000000008:001     0    0/   0     (006)"003030343135"
5  62 0062:0000000018:002     1 0032/0032
5  62 0065:0000000018:001     0    0/   0     (006)"003030343335"
5  62 0062:0000000021:002     1 0032/0032
5  62 0065:0000000021:001     0    0/   0     (006)"003030343035"
5  62 0062:0000000026:002     1 0032/0032
5  62 0065:0000000026:001     0    0/   0     (006)"003030343138"
5  62 0062:0000000032:002     1 0032/0032
5  62 0065:0000000032:001     0    0/   0     (006)"003030343731"
5  62 0062:0000000046:002     1 0032/0032
5  62 0065:0000000046:001     0    0/   0     (006)"003030343136"
0-------------------------------------------------------------------------------
0
0 Indices for database  - $DUA0:[ORION]MF_PERSONNEL.RDB;
0
0-------------------------------------------------------------------------------
0 Hashed index EMPLOYEES_HASH for relation EMPLOYEES duplicates not allowed
0 Max Level: 1, Nodes: 69, Used/Avail: 2797/2797 (100%), Keys: 100, Records: 100
0
0     Level: 1, Nodes: 69, Used/Avail: 2797/2797 (100%), Keys: 100, Records: 100
0
0-------------------------------------------------------------------------------
```

RMU Analyze Indexes コマンドで Option=Debug 修飾子を指定すると、インデックス・ノードとインデックス・レコードに関する詳細情報が表示されます。出力のヘッダーは、概要と詳細の 2 つの部分で構成されます。ヘッダーの後に、詳細なインデックス・レコード情報が表示されます。出力の最後の部分には、RMU Analyze Indexes コマンドに Option=Full

修飾子を指定した場合と同じ情報が表示されます。ただし、インデックスが複数の論理領域に格納される場合は、すべての論理領域の情報が表示されます。

次のリストで、RMU Analyze Indexes Option=Debug による表示について説明します。各フィールドの説明の後にあるカッコ内のエントリは、EMPLOYEES_HASH インデックスを使用した例 3-45（3-102）に対応します。この説明では、論理領域 58 と、この論理領域の詳細なインデックス情報の最初の行を参照します。

- インデックス情報の概要
  - ID
    インデックスの論理領域 ID（58）。
  - Hash
    コード化されたフィールド。ハッシュ・インデックスの場合は T = TRUE、ソート・インデックスの場合は F = FALSE（T）
  - Dup
    コード化されたフィールド。重複を許可する場合は T = TRUE、重複を許可しない場合は F = FALSE（T）
  - Name
    インデックスの名前（EMPLOYEES_HASH）。
  - Relation
    インデックスを定義するテーブルの名前（EMPLOYEES）。
- インデックス情報の詳細
  - ID
    インデックスの論理領域 ID（58）。
  - DB KEY
    レコードの dbkey（0058:0000000002:002）。論理領域 ID（0058）、ページ番号（0000000002）、レコードを格納するページ上の行番号（002）の 3 つの部分で構成されます。
  - LEVEL
    インデックス・レコードのレベル（1）。ノード・レコードまたはハッシュ・バケットを表します（この場合はハッシュ・バケット）。
  - SIZE
    ハッシュ・インデックスでは各ハッシュ・バケットの合計サイズ（バイト数）の後にスラッシュ（/）が続き、その後にハッシュ・バケットに割り当てられた領域のサイズ（バイト数）が続きます（0051 バイト /0051 バイト）。
    ソート・インデックスでは（例 3-46（3-106）を参照）、すべてのインデックス・レコードに使用する領域の合計サイズ（バイト数）の後にスラッシュ（/）、さらに

ノード・レコードに割り当てられた領域のサイズ（バイト数）が続きます（0051 バイト /0051 バイト）。

- KEY-LEN
  キーの長さ（バイト数）(6)。
  ソート・インデックスでは（例 3-46（3-106）を参照）、カッコ内の値に次の意味があります。先頭の 3 桁（接頭辞）は、キー値を前のキー値と比較した場合に前のキーと同じバイト数を表します。プラス記号（+）の後に続く次の 3 桁（接尾辞）は、前のキーと異なるバイト数を表します。
  たとえば、KEY-LEN（000+005）で、000 は前のキーと同じバイトがないことを表し（前のキーがないので）、5 バイトが異なることを表します。次の行では KEY-LEN が（001+004）になっています。値 001 は最初のバイト（00）が前のキーと同じで、残りの 4 バイトが異なることを表します。次の行では KEY-LEN が（003+002）になっています。値 003 は最初の 3 バイト（00454C）が前のキーと同じで、残りの 2 バイトが異なることを表します。
  この情報は、接頭辞が繰り返されるのでこれを圧縮する場合のキー圧縮に有効です。B ツリー・インデックスをたどる場合は、リーフ・ノードに達するまでツリーのブランチごとに接頭辞をピック・アップします。リーフ・ノードでは接頭辞の値が一意です。DEPARTMENTS_INDEX ソート・インデックスでは、KEY-LEN の合計は 5 バイトです。

- KEY
  実際のキーが 16 進で表示されます（003030313635）。

EMPLOYEES_HASH ハッシュ・インデックスに関する例 3-45（3-102）で、テーブル EMPLOYEES のインデックスが 3 つの論理領域（58、60、62）に分割されていることに注意してください。論理領域ごと、論理領域内のページ番号ごとに、dbkey がリスト・アップされています。キーのサイズは 5 バイトで、1 バイトの NULL ビット・ベクトルと合わせて 6 バイト、さらに 1 バイトのキー・サイズを加えて合計 7 バイトになります。個々のハッシュ・バケットの合計サイズが表示されています。たとえば、最初のハッシュ・バケットは 0051 です。このハッシュ・バケットの 51 バイトには、13 バイトの固定ヘッダー情報が含まれます。また、ハッシュ・バケットに伴うデータ行には、それぞれ 19 バイトの情報があります。19 バイトの内容は次のとおりです。

- キーが 7 バイト。長さ +NULL ビット・ベクトル + 行のキー・サイズ
- 行の重複件数が 4 バイト
- 行（または重複を許可する場合は重複ノード）の dbkey ポインタが 8 バイト

したがって、ハッシュ・バケットは 19 バイトずつ増えることになります。このインデックスの最小のハッシュ・バケットは 32 バイトです（1 データ行分の情報 19 バイト + ハッシュ・バケットの固定ヘッダー情報 13 バイト）。このハッシュ・バケットは 2 つのデータ行を伴うので、ハッシュ・バケットのサイズは 51 バイトです。データ行ごとに、実際の dbkey とキー値が表示されます。

RMU Analyze Indexes コマンドで Option=Debug 修飾子と DEPARTMENTS_INDEX ソート・インデックスを指定した場合は、例 3-46（3-106）に示す情報が表示されます。

例 3-46 RMU Analyze Indexes Option=Debug コマンド（ソート・インデックスの場合）

```
$ RMU/ANALYZE/INDEXES mf_personnel DEPARTMENTS_INDEX /OPTION=DEBUG
0------------------------------------------------------------------------------
0
0 Indices for database  - $DUA0:[ORION]MF_PERSONNEL.RDB;
0
0   ID   Hash Dup Name                              Relation
0------------------------------------------------------------------------------
4      72   F   F  DEPARTMENTS_INDEX               DEPARTMENTS
0
0------------------------------------------------------------------------------
0
0   ID        DB KEY       LEVEL    SIZE    KEY-LEN    KEY
0------------------------------------------------------------------------------
0
5  72 0072:0000000002:002     1 0188/0430     (000)""
5  72 0073:0000000002:001     0    0/   0 (000+005)"0041444D4E"
5  72 0073:0000000002:003     0    0/   0 (001+004)"00454C454C"
5  72 0073:0000000002:004     0    0/   0 (003+002)"00454C4753"
5  72 0073:0000000002:005     0    0/   0 (003+002)"00454C4D43"
5  72 0073:0000000002:006     0    0/   0 (002+003)"00454E4720"
5  72 0073:0000000002:007     0    0/   0 (001+004)"004D424D46"
5  72 0073:0000000002:008     0    0/   0 (004+001)"004D424D4E"
5  72 0073:0000000002:009     0    0/   0 (004+001)"004D424D53"
5  72 0073:0000000003:001     0    0/   0 (002+003)"004D43424D"
5  72 0073:0000000003:002     0    0/   0 (004+001)"004D434253"
5  72 0073:0000000003:003     0    0/   0 (002+003)"004D475654"
5  72 0073:0000000003:004     0    0/   0 (002+003)"004D4B5447"
5  72 0073:0000000003:005     0    0/   0 (002+003)"004D4E4647"
5  72 0073:0000000003:006     0    0/   0 (002+003)"004D534349"
5  72 0073:0000000003:007     0    0/   0 (003+002)"004D534D47"
5  72 0073:0000000003:008     0    0/   0 (002+003)"004D54454C"
5  72 0073:0000000003:009     0    0/   0 (001+004)"005045524C"
5  72 0073:0000000003:010     0    0/   0 (004+001)"0050455253"
5  72 0073:0000000003:011     0    0/   0 (002+003)"005048524E"
5  72 0073:0000000003:012     0    0/   0 (002+003)"0050524D47"
5  72 0073:0000000003:013     0    0/   0 (001+004)"0053414C45"
5  72 0073:0000000003:014     0    0/   0 (002+003)"0053455552"
5  72 0073:0000000003:015     0    0/   0 (002+003)"0053554E45"
5  72 0073:0000000003:016     0    0/   0 (003+002)"0053555341"
5  72 0073:0000000003:017     0    0/   0 (004+001)"005355534F"
5  72 0073:0000000002:010     0    0/   0 (003+002)"0053555745"
0------------------------------------------------------------------------------
0
```

```
0 Indices for database  - $DUA0:[ORION]MF_PERSONNEL.RDB;
0
0-------------------------------------------------------------------------------
0 Index DEPARTMENTS_INDEX for relation DEPARTMENTS duplicates not allowed
0 Max Level: 1, Nodes: 1, Used/Avail: 188/398 (47%), Keys: 26, Records: 26
0
0     Level: 1, Nodes: 1, Used/Avail: 188/398 (47%), Keys: 26, Records: 26
0
0-------------------------------------------------------------------------------
```

例 3-46（3-106）に、DEPARTMENTS テーブルの DEPARTMENTS_INDEX ソート・インデックスを示します。このインデックスは、ランクなしソート・インデックスです。

インデックス全体（26 レコード）は論理領域 72 のページ 2 とページ 3 上に存在し、使用可能な 430 バイトのうち 188 バイトを使用しています。使用率は 47% です。インデックスを圧縮すると、ノード・サイズが 422 バイトから 188 バイトに減少し、ノード・レコードの領域使用率が 98% から 47% に低下していることがわかります。出力の終わりに表示されるサマリー情報の Used/Avail の値にインデックスのヘッダーとトレーラの情報がないことにも注意してください。この情報は 32 バイトに相当します。この値は、ノード・レコードごとに出力の詳細部分に表示されます。インデックスで使用するバイト数は、次のように計算します。ソート・キーは、4 バイトと NULL バイトで合計 5 バイトです。接頭辞が 1 バイト、接尾辞が 1 バイトです。接頭辞は前のキーと同じバイトの数を表し、接尾辞は前のキーと異なるバイトの数を表します。行への dbkey ポインタは 8 バイトです。26 のデータ行に 15 バイトをかけると合計 390 バイトになります。15 バイトの内容は次のとおりです。

- ソート・キーが 7 バイト。長さ +NULL バイト + 接頭辞 + 接尾辞
- 行への dbkey ポインタが 8 バイト。

390 バイトにインデックス・ノードのヘッダーとトレーラの情報の 32 バイトを加えて合計 422 バイトを使用します。インデックスを圧縮すると、使用するバイト数が 188 バイトに削減されます。

ランク付きとランクなしのソート・インデックスの詳細は、『Oracle Rdb7 Guide to Database Design and Definition』を参照してください。

ソート・インデックスでは、インデックス・ノードの長さと重複の件数（許可される場合）に特に注意する必要があります。インデックスが長い場合は、テーブル上のインデックスを保持するのに多くのシステム・リソースを使用します。同時に、重複が許可され、インデックスに多くの重複値が存在する場合は、インデックスの効率が低下します。インデックスを使用した場合にインデックスの格納と更新に必要なオーバーヘッドを正当化するに足るだけのパフォーマンスの改善があるかどうかを判断するには、さらにテストを行う必要があります。インデックスを使用すると、マルチセグメントのインデックスを定義して現在使用できる以外のインデックス・ノードにアクセスを分散できるという利点もあります。

例 3-47（3-108）に、RMU Show Indexes Option=Debug コマンドで圧縮型インデックスを指定した場合の出力の一部を示します。

例 3-47 RMU Analyze Indexes Option=Debug コマンド（圧縮型インデックスの場合）

```
$ RMU/ANALYZE/INDEXES test_db COMPRESSED_IND /OPTION=DEBUG
0------------------------------------------------------------------------------
0
0 Indices for database  - $DUA3:[ORION]TEST_DB;
0
0   ID   Hash Dup Name                            Relation
0------------------------------------------------------------------------------
4      13   F   T  COMPRESSED_IND                  NEW_TABLE
0
0------------------------------------------------------------------------------
0
0   ID        DB KEY       LEVEL    SIZE    KEY-LEN    KEY
0------------------------------------------------------------------------------
0
5  13 0004:0000000146:005     0    0/   0  (006+025)"0052444224524C435F434F4E535
5241494E545F4E414D455F4E445820202020"
5  13 0004:0000000147:001     0    0/   0  (009+017)"0052444224524C435F4649454C4
5F4E414D455F4E44582020202020202020"        ^          ^
        .                                  |          |
        .                                  |          Hexadecimal dumps
        .                                  |          (uncompressed keys)
                                           These lengths are for the
                                           compressed index key
```

RMU Analyze Indexes Option=Debug コマンドでは、圧縮型インデックスのインデックス・キーを非圧縮型に戻してから表示します。16 進の出力は非圧縮型インデックス・キーを表しています。ただし、例 3-47（3-108）で個々のインデックス・キーについて報告される長さは圧縮型インデックス・キーの長さです。

3.9.5.2 Performance Monitor によるインデックス情報の収集

Performance Monitor では、次の画面でソート・インデックスとハッシュ・インデックスに関する情報を収集できます。

- 「Hash Index Statistics」画面
  「Hash Index Statistics」画面では、データベースのハッシュ・インデックスの更新および検索アクティビティを監視します。この画面には、キーの挿入と削除の合計数が表示されます。開始されたスキャンの数も示されます。検索（正常なフェッチ）では、フェッチされたノード（パケット断片または重複ノード）の合計数が表示されます。ハッシュ・インデックスの分析に関するアドバイスは、8.1.3.4 項（8-29）と 8.1.3.5 項（8-33）にもあります。

- 「Index Statistics (Retrieval)」画面
  「Index Statistics (Retrieval)」画面では、データベースのソート・インデックスで発生する検索アクティビティの量を監視します。Oracle Rdb では、多くの場合、ダイレクト・

インデックス参照とインデックス・スキャンを使用してデータベース内のレコードにアクセスします。「Index Statistics (Retrieval)」画面では、これらの操作とフェッチされたインデックス・ノードの数を監視します。

- 「Index Statistics (Insertion)」画面
  「Index Statistics (Insertion)」画面では、挿入時、すなわちインデックス・キー・フィールドの挿入または変更を行う場合、あるいはテーブルに関して SQL の CREATE INDEX 文を使用する場合のデータベースのソート・インデックスの更新アクティビティを監視します。この画面には、挿入が発生するインデックス・ノードのタイプとノード・タイプごとのノード作成も表示されます。この画面を調べると、データベースへの挿入発生後にデータベースがそのソート・インデックスを調整する方法を監視できます。

- 「Index Statistics (Removal)」画面
  「Index Statistics (Removal)」画面では、削除操作を実行した場合、すなわちインデックス・キー・フィールドの削除または変更を行う場合、あるいはインデックスを削除する場合のデータベースのソート・インデックスの更新アクティビティを監視します。この画面には、削除が発生するインデックス・ノードのタイプが表示されます。ノード・タイプごとのノードの削除も表示されます。この画面では、インデックスからノードが削除された場合にデータベースがそのソート・インデックスを調整する方法を監視できます。

各画面と画面上の各フィールドの説明については、Performance Monitor のヘルプを参照してください。

3.9.6 ソート・インデックス構造

ソート・インデックス構造はツリー構造に基づいており、複数のノードで構成されています。個々のソート・インデックス（システムまたはユーザー）には、独自のインデックス構造または B ツリー構造があります。個々の B ツリー構造は、バランスがとれた階層構造内でインデックス・ノードをリンクして作成されます。これらのノードは、キー値の下位から上位に向かって、水平方向にもリンクされます。ノード間のリンクは、dbkey を使用して作成されます。

B ツリーのノード・タイプは、ソート・インデックスがランク付きかランクなしかによって変わります。

ランク付きソート・インデックスには、次のノード・タイプがあります。

- インデックス・キー・ノード
  インデックス・キー・ノードには、一意インデックス・キーと他のノードへのポインタがあります。B ツリー構造では、複数レベルのインデックス・キー・ノードを使用できます。レベル 1 のノードは**リーフ・ノード**としても知られており、テーブル内の行かオーバーフロー・ノードを指します。レベル 2（以上）のノードでは下位レベルのインデックス・ノードを指します。
  ランク付きソート・インデックスでは、Oracle Rdb はバイト単位のビットマップ圧縮を使用して重複を圧縮します。この場合は、インデックス値の同じ複数のデータ行を指す dbkey のリストを使用して、圧縮型ビットマップとしてリーフ・ノードに格納します。

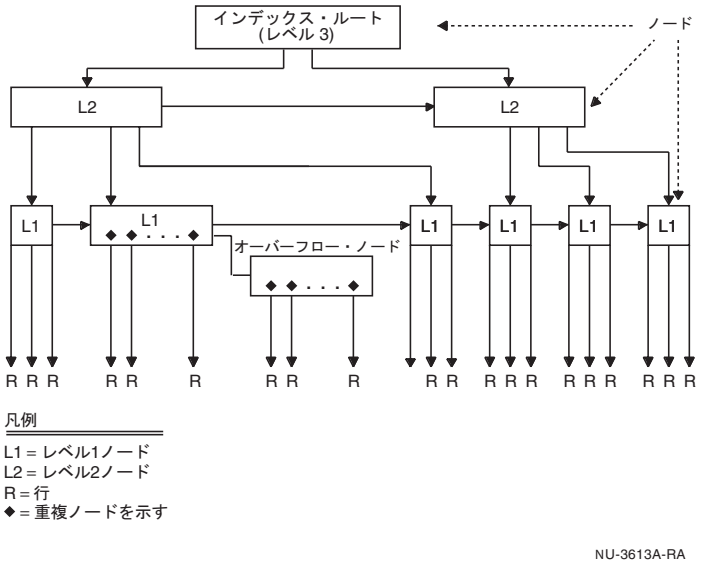
重複エントリを格納する場合は、レベル 1 のノードに dbkey の圧縮型ビットマップが入ります。

- オーバーフロー・ノード
  Oracle Rdb では、リーフ・ノード内の圧縮型ビットマップがオーバーフローすると、オーバーフロー・ノードが作成されます。**オーバーフロー・ノード**には、リーフ・ノードに格納された圧縮型ビットマップの続きと次のオーバーフロー・ノードへのポインタが入ります。

インデックス・ノードはそれ自体が行なので、行ロックの影響を受けます。

図 3-6（3-110）に、ランク付きソート・インデックスの B ツリー・インデックス構造を示します。

図 3-6 ランク付きソート・インデックスの B ツリー・インデックス構造



レベル 2（以上）のノードの内容は次のとおりです。

- レコード・タイプ・ヘッダー。インデックス・ノードかオーバーフロー・ノードかを表します。
- 2 つの断片化フラグ。
- 記憶領域 ID。インデックス・ノードの論理領域を特定します。
- レベル・タイプ。インデックス・ノードが存在する B ツリー構造のレベルを表します。
- 接頭辞と接尾辞の圧縮によるインデックス・キー値。

- 行のエントリ・カーディナリティとリーフ・カーディナリティ。
- 圧縮型ビットマップとして格納された dbkey のリスト。これらの dbkey は、インデックス値の同じデータ行を指しています。
- 同じレベルでキー値がより大きい右隣のノードへのポインタ。
- 下位レベルのノードへのポインタ。

レベル 1 のノード（データ行に最も近いレベル）の内容は次のとおりです。

- レコード・タイプ・ヘッダー。インデックス・ノードかオーバーフロー・ノードかを表します。
- 2 つの断片化フラグ。
- 記憶領域 ID。インデックス・ノードの論理領域を特定します。
- レベル・タイプ。インデックス・ノードが存在する B ツリー構造のレベルを表します。
- 接頭辞の圧縮によるインデックス・キー値（キー値の先頭バイトが前のエントリと同じ場合は格納しない）
- ノードに重複がある場合、データ行へのポインタ。行の相対的な位置を格納する圧縮型 dbkey を使用します。重複があっても、重複するまでポインタは使用されません。オーバーフローが発生した時点で、ポインタはオーバーフロー dbkey の位置を示します。
- 行のエントリ・カーディナリティ。
- ノードに重複がある場合は、圧縮型ビットマップとして格納される dbkey のリスト。これらの dbkey は、インデックス値の同じデータ行を指しています。
- 必要な場合はオーバーフロー・ノードへのポインタ（Oracle Rdb では、重複のリストがインデックス・ノードをオーバーフローすると、オーバーフロー・ノードが作成されます）。
- 右隣のレベル 1 ノード（キー値はより大きい）へのポインタ。

オーバーフロー・ノードの内容は次のとおりです。

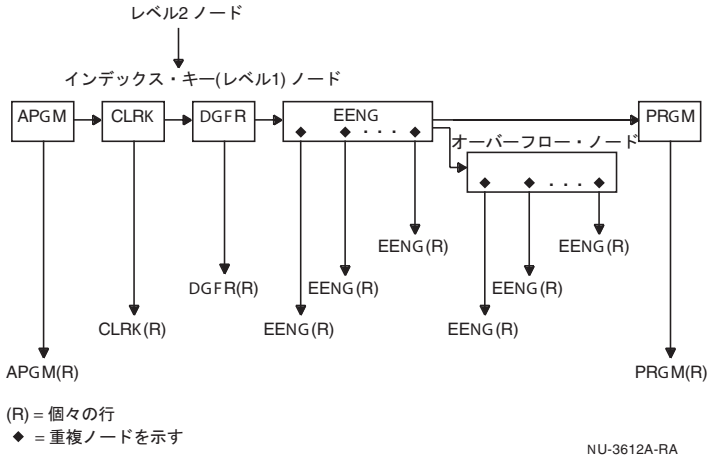
- レコード・タイプ・ヘッダー。インデックス・ノードか重複ノードかを表します。
- 2 つの断片化フラグ。
- 記憶領域 ID。インデックス・ノードの論理領域を特定します。
- 圧縮型ビットマップとして格納された dbkey のリスト。これらの dbkey は、インデックス値の同じデータ行を指しています。
- 連鎖内の次のオーバーフロー・ノードへのポインタ。

重複レコードでは、個々のレコードを指すかわりに、インデックス・キー値の同じすべてのデータ行を次々に指す dbkey のリストがエントリに含まれます。多くのデータ行のインデッ

クス・キー値が同じ場合は、1 つ以上のオーバーフロー・ノードが必要になることがあります。これは、オーバーフロー連鎖と呼ばれます。

図 3-7（3-112）に、オーバーフロー・ノードのある B ツリー構造を示します。

図 3-7 オーバーフロー・インデックス・ノード



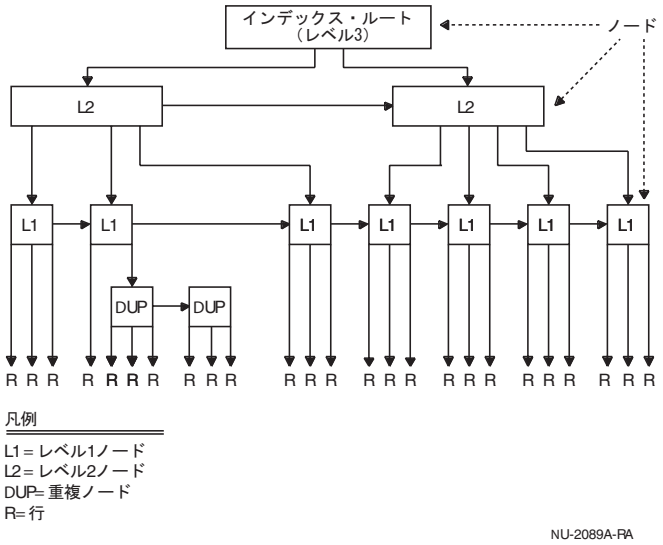
ランクなしソート・インデックスには、次のノード・タイプがあります。

- インデックス・キー・ノード
  インデックス・キー・ノードには、一意インデックス・キーと他のノードへのポインタがあります。B ツリー構造では、複数レベルのインデックス・キー・ノードを使用できます。レベル 1 ノードはリーフ・ノードとしても知られており、テーブル内の行か重複インデックス行を指します。レベル 2（以上）のノードでは、下位レベルのインデックス・ノードを指します。
- 重複ノード
  Oracle Rdb では、UNIQUE キーワードを使用しないでインデックスを定義し、重複するキー値を格納した場合に重複インデックス・ノードが作成されます。重複ノードには、重複する行 dbkey のリストが含まれます。

インデックス・ノードはそれ自体が行なので、行ロックの影響を受けます。

図 3-8（3-113）に、ランクなしソート・インデックスの B ツリー・インデックス構造を示します。

図 3-8 ランクなしソート・インデックスの B ツリー・インデックス構造



レベル 2（以上）のノードの内容は次のとおりです。

- レコード・タイプ・ヘッダー。インデックス・ノードか重複ノードかを表します。
- 2 つの断片化フラグ
- 記憶領域 ID。インデックス・ノードの論理領域を特定します。
- レベル・タイプ。インデックス・ノードが存在する B ツリー構造のレベルを表します。
- 接頭辞と接尾辞の圧縮によるインデックス・キー値
- 同じレベルでキー値のより大きい右隣のノードへのポインタ
- 下位レベルのノードへのポインタ

レベル 1 のノード（データ行に最も近いレベル）の内容は次のとおりです。

- レコード・タイプ・ヘッダー。インデックス・ノードか重複ノードかを表します。
- 2 つの断片化フラグ
- 記憶領域 ID。インデックス・ノードの論理領域を特定します。
- レベル・タイプ。インデックス・ノードが存在する B ツリー構造のレベルを表します。

- 接頭辞の圧縮によるインデックス・キー値（キー値の先頭バイトが前のエントリと同じ場合は格納しない）
- データ行へのポインタ。行の相対的な位置を格納する圧縮型 dbkey を使用します。
- 重複ノードへのポインタ。論理領域番号が負の dbkey を使用します。
- 右隣のレベル 1 ノード（キー値はより大きい）へのポインタ

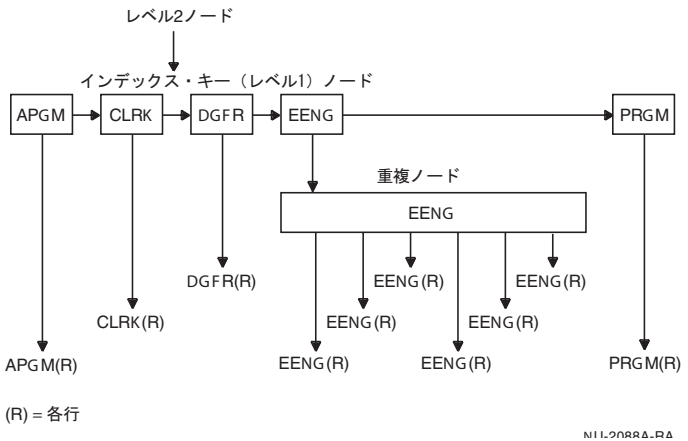
重複ノードの内容は次のとおりです。

- レコード・タイプ・ヘッダー。インデックス・ノードか重複ノードかを表します。
- 2 つの断片化フラグ
- 記憶領域 ID。インデックス・ノードの論理領域を特定します。
- インデックス・キー値の同じデータ行へのポインタ
- 重複連鎖内の次の重複ノードへのポインタ

重複ノード・レコードは、個々のレコードを指すかわりに、B ツリー構造によってインデックス・キー値の同じすべてのデータ行を次々に指す重複ノードを指します。多くのデータ行のインデックス・キー値が同じ場合は、1 つ以上の重複ノードが必要になることがあります。これは、**重複連鎖**と呼ばれます。

図 3-9（3-114）に重複ノードを示します。

図 3-9 重複インデックス・ノード

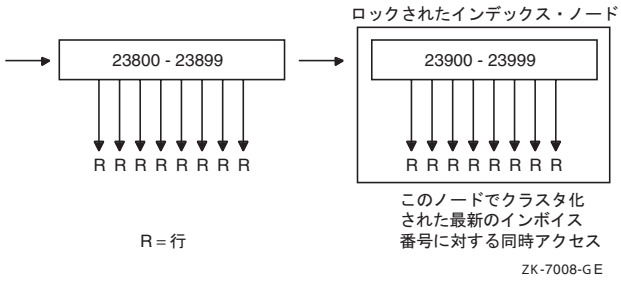


3.9.6.1 時系列キーのロックの削減

インデックス構造におけるロックの競合を増大するインデックス設計の 1 つの例として、**時系列キー**があります。時系列キーでは、データベースに格納されるインデックス・キー値は時間の経過とともに値を増します。したがって、キー値へのアクセスはインデックス構造に属する少数の同じノード内だけで行われる傾向があります。

この種のキーのよい例は INVOICE_NUMBER 列です。最新のインボイス番号は古いインボイス番号よりも大きく、インボイスへのアクセスは最新の番号の追加に集中する傾向があります。このようなインデックスでは、アクセスの多くが少数のノードに集中するので、図 3-10（3-115）に示すように、他のユーザーがテーブルにアクセスしている間にインデックスを更新すると、データベースの並行性が制限される場合があります。

図 3-10 1 つのインデックス・ノードへの集中的なアクセス



このロックの競合の問題には、複数のソリューションがあります。

- インボイス番号に別のキー値を連結します。これは、通常はクエリーを満足するのに必要な列を使用します。目標は、特定の部分にアクセスを集中することなく、マルチセグメント・キーの値をインデックス全体により均等に配分するようなキーの値を設定することです。INVOICE_NUMBER 列の場合は、CUSTOMER_NUMBER 列と INVOICE_NUMBER 列を含むマルチセグメント・インデックスが適しています。例 3-48（3-115）に示す SQL の CREATE INDEX 文では、このマルチセグメント・インデックスを作成しています。このアプローチの欠点は、新しい冗長なデータがインデックス構造に格納され、追加の記憶領域を保持する必要があることです。

  例 3-48 時系列キーによるロックの競合の削減

  ```
  SQL> CREATE UNIQUE INDEX CUSTOMER_INVOICE ON INVOICES
    1>   (CUSTOMER_NUMBER,
    2>   INVOICE_NUMBER);
  ```

- 夜間のバッチで時系列キーを事前に割り当ててから、オンライン操作時に実際のデータで行を変更します。これでインデックスの競合は回避されますが、断片化が発生する恐れがあります。

- ハッシュ・インデックスを使用できるかどうかを検討します。ハッシュ・インデックスでは時系列キーの問題がありません。

3.9.6.2 重複ノード・キーのロックの削減

ロックの競合に関するもう 1 つの問題は、テーブルに関する非常に少数のキー値に対して重複が非常に多く、更新が頻繁に行われる場合に発生します。1 つのインデックス・ノードで多くのキー値を保持できるので、わずか 1 つのキーを更新するのにノード全体がロックされる可能性があります。

この重複の問題を示す一般的な例は、2 文字の STATE コードに関して定義したインデックスとインデックス・ノードごとに多くの重複のあるメーリング・リストです。インデックスの最上位ノードにすべてのキー値を含めることができます。重複値の 1 つを更新する場合は、更新トランザクションで最上位ノード（ツリーのルート）をロックし、他のユーザーによるテーブル・アクセスを拒否します。

この問題のソリューションは、STATE コード列に POSTAL_CODE 列を連結することです。このマルチセグメント・キーによって、より規模の大きい広範なインデックス構造にキー値を分散すると、ロックの競合を削減するのに役立ちます。欠点は、より大きなインデックス構造を保持するためにより大きな記憶領域が必要になることです。

次に示す SQL の CREATE INDEX 文と RMU Analyze Indexes コマンドの出力では、mf_personnel データベースの EMPLOYEES テーブルで STATE 列にインデックスが定義されていることを表します。第 2 のマルチセグメント・インデックスも STATE と POSTAL_CODE の 2 列を使用して定義されており、より多くのインデックス・ノードに値を分散しています（例 3-49 （3-117）を参照）。

例 3-49 多くの重複によるロックの競合の削減

```
SQL> CREATE INDEX STATE_IDX ON EMPLOYEES
  1> (STATE)
  2> TYPE IS SORTED;
SQL>
SQL> CREATE INDEX STATE_POSTAL_IDX ON EMPLOYEES
  1> (STATE,
  2> POSTAL_CODE)
  3> TYPE IS SORTED;
SQL> COMMIT;
SQL> EXIT
$ RMU/ANALYZE/INDEXES mf_personnel STATE_IDX, STATE_POSTAL_IDX
-------------------------------------------------------------------------------
 Indices for database  - $DUA0:[ORION]MF_PERSONNEL.RDB;
-------------------------------------------------------------------------------
 Index STATE_IDX for relation EMPLOYEES duplicates allowed
 Max Level: 1, Nodes: 1, Used/Avail: 25/398 (6%), Keys: 3, Records: 1
     Duplicate nodes: 3, Used/Avail: 792/868 (91%), Keys: 2, Records: 99
-------------------------------------------------------------------------------
 Index STATE_POSTAL_IDX for relation EMPLOYEES duplicates allowed
 Max Level: 2, Nodes: 3, Used/Avail: 314/1194 (26%), Keys: 35, Records: 9
     Duplicate nodes: 24, Used/Avail: 728/1392 (52%), Keys: 24, Records: 91
-------------------------------------------------------------------------------
```

インデックス構造は EMPLOYEES テーブルへの同時アクセスを増加できるように設計されていますが、アクセスは 3 つの重複ノードに集中しており、インデックスの大部分にはアクセスされません。重複値がある列のインデックスを定義するときに、テーブルには多くの行があるのにインデックスを定義した列の値はわずかしかないことに気付く場合があります。この場合、STATE_POSTAL_IDX を使用すると、他のトランザクションが EMPLOYEES テーブルにアクセスし、このトランザクションでロックしていない行を検索できるので便利です。

このインデックス付けの欠点は、Oracle Rdb ではマルチセグメント・インデックス・キーの値を分散するにはより大きな記憶領域が必要になることです。ただし、1 つ以上の列にインデックスを定義しても、特定のタスクにインデックスを使用してインデックスが不要になった場合はこれを削除できます。たとえば、一時的にレポートを作成するためにインデックスを使用できます。読取り専用のレポートに必要なインデックスを作成し、レポートを実行した後に削除してください。それからデータベースを本番で使用します。こうして、同時アクセスの増大による恩恵を受けながら、ディスクを効率的に使用します。

3.9.6.3 クラスタ化インデックス

クラスタ化インデックスは、記憶領域に事前にソートしたテーブル行を配置するために使用するソート・インデックスです。この記憶領域には、インデックスとほぼ同じ順序で行が格納されます。これで、範囲検索を必要とするクエリーのパフォーマンスを大幅に改善できます。たとえば、LAST_NAME 列で昇順にインデックスを作成してから、SQL の CREATE

STORAGE MAP 文の PLACEMENT VIA INDEX 句でこのソート・インデックスを使用してテーブルを格納するように指定すると、Oracle Rdb はアルファベット順に行を格納しようとします。何も指定しない場合は、ストレージ・マップでなくソート・インデックスを使用して、新しい記憶領域を定義し、記憶領域にテーブルの行を配置するので、行はランダムに格納されます。

インデックスを定義する場合は、Oracle Rdb がインデックスを配置する記憶領域を決定するためのキーとして使用するインデックス内の列のリストを指定できます。SQL で CREATE INDEX 文の STORE USING 句の WITH LIMIT OF オプションを使用すると、指定した記憶領域に最初に格納するときのキーの最大値を設定できます。STORE USING 句でリストに列名を指定する順序は、インデックス内の順序と同じでなければなりません。ストレージ・マップを定義する場合は、記憶領域に行を配置するために使用するインデックスの名前を指定します。行を複数の記憶領域に格納する場合は、SQL で CREATE STORAGE MAP 文の STORE 句の WITH LIMIT OF オプションを使用すると、指定した記憶領域に最初に格納するときのキーの最大値を設定します。リスト内の列名は、定義済みのインデックスと同じ順序でなければなりません。この文の構文と詳細は、『Oracle Rdb7 Guide to Database Design and Definition』と『Oracle Rdb7 SQL Reference Manual』を参照してください。

クラスタ化インデックスは、範囲検索を要求するクエリーのパフォーマンスを大きく左右する可能性があります。LAST_NAME 列に基づくインデックスを使用してテーブルのクラスタ化を定義し、姓が Smith より大きい従業員をすべて検出するクエリーを記述したと仮定します。姓が Smith より大きい従業員が 100 人いて 1 ページに 5 人の従業員が入る場合、クラスタ化インデックスを使用すると Oracle Rdb は約 20 回の I/O 操作でクエリーに回答します。非クラスタ化インデックスを使用した場合、Oracle Rdb では 100 回の I/O 操作が必要です。

3.9.6.4 ソート・インデックスによる時間の経過に伴うパフォーマンス低下の回避

ソート・インデックスを使用して記憶領域に行を格納する場合、各行はインデックス・ノード・レコード内のポインタ・レコードに近接するインデックス・キー・シーケンスに格納されます。格納は、記憶領域内で領域管理（SPAM）ページの直後のページから始まります。各ページに空きがなくなると、次に使用可能なページを使用して行を格納します（以下同様）。インデックス・ノード・レコードに空きがなくなると、次に使用可能なページの新しいインデックス・ノード・レコードに書込みを行います。オリジナルのノード・レコードにはフォワード・ポインタが書き込まれ、新しいインデックス・ノード・レコードに連鎖します。こうして、各行はそのポインタ・レコードに近接するインデックス・キー・シーケンスの記憶領域に格納され、記憶領域の最初からテーブル行とインデックス・ノード・レコードをすべて格納するまで続きます。

データをロードした直後は、範囲検索のパフォーマンスが最適になります。デフォルトまたは指定のチューニング・パラメータの下で、行とそのポインタ・レコードが互いにできるだけ近接する場所に配置されるためです。ページ・サイズ、ノード・サイズや使用率などのインデックス・パラメータ、バッファ・サイズはアプリケーションの検索パフォーマンスを最

適化するようにチューニングされていると仮定した場合、1 回から数回の I/O 操作で行の範囲検索を実行できます。

初期データ・ロード操作後に、データベースに新しい行が挿入され、インデックス・キー列データの値が何度も更新されると、初めて範囲検索パフォーマンスの緩やかな低下が心配されます。理由は単純です。新しい行は空き領域のある次のページに格納されます。通常は、記憶領域内で空きのない最後のページの後になります。

しかし、インデックス行ポインタ・エントリは新しいインデックス・キー列の値を格納した個々のインデックス・ノード・レコードに挿入されます。この結果、行とインデックス・ポインタ・レコード・ページのずれは初期データ・ロード操作で通常発生するものより大きくなります。さらに、インデックス・ノードに行ポインタが追加されると、ノード・レコードは強制的に分割され、再調整されます。新しいノード・レコードは記憶領域内で空きのない最後のページの次のページに書き込まれ、既存の行からのずれはさらに大きくなります。行が更新されると、行が移動することはありませんが、行のインデックス・キー列の値が変わるとポインタ・レコードが更新され、移動します。すなわち、ポインタ・レコードは適切なノード・レコードに移動し、インデックス・キー・シーケンスを保持します。再びノード・レコードが分割され再調整されると、行とそのノード・レコード・ポインタは互いにどんどん離れていく場合があります。したがって、ページのずれは新しい行の追加や既存の行の更新によって発生する可能性があります。

この個々のノード・ポインタ・レコードとその行との物理的なページのずれによって、検索パフォーマンスが低下することがあります。時間が経過するにつれて多くの新しいデータが追加され、キー・インデックス列の値が変更されるので、指定したインデックス・キー値と対応する行の範囲検索を行ってバッファに格納するにはより多くの I/O 操作が必要になります。ついには、パフォーマンスが低下することがあります。

この問題を回避するために、記憶領域内のデータを時折アンロードしてから再ロードし、個々のインデックス・ノード・レコード・ポインタとその行とのページのずれを最小化してください。データを再ロードする前に、インデックスと記憶領域を再調査し、さらにチューニングを実行して検索パフォーマンスを最適化できるかどうかを確認します。

3.9.6.5 ソート・インデックスによる前方スキャンと後方スキャン

Oracle Rdb は、1 つのソート・インデックスに関して前方スキャンと後方スキャンを実行できるので、標準のソート順でも逆順でも検索できます。

後方スキャンによって一連の重複レコード（インデックス・キー値の同じレコード）の dbkey とインデックス・キーを返す場合、重複レコードの dbkey は前方スキャンによる場合と同じ順序で返されます。これで、重複キーの後方スキャンの効率が大幅に向上します。

ソート・インデックスではインデックス・キーの同じレコード間で特定の順序が保証されないので、この最適化は意味論的にも容認できます。

前方と後方の両方のインデックス・スキャンを実行するために 2 つのインデックスは必要ありませんが、後方スキャンは通常の前方スキャンに比べて多くの CPU リソースを使用する場合があります。

例 3-50（3-120）では、RDMS$DEBUG_FLAGS が "S" に設定されているので、クエリーを実行するとオプティマイザが選択したクエリー・ストラテジが表示されます。2 番目の文の RDMS$DEBUG_FLAGS 出力に注目すると、後方スキャンを使用してクエリーを処理することを示しています。

例 3-50 1 つのソート・インデックスによる前方スキャンと後方スキャン

```
SQL> -- The optimizer uses forward scan for the LAST_NAME index
SQL> -- (an ascending index) when the values are requested in the
SQL> -- same order as the index (ascending).
SQL> SELECT LAST_NAME, FIRST_NAME FROM EMPLOYEES
  1> WHERE LAST_NAME > 'A' AND LAST_NAME < 'M'
SQL> ORDER BY LAST_NAME ASCENDING;
Conjunct        Get     Retrieval by index of relation EMPLOYEES
  Index name  EMP_LAST_NAME [1:1]
 LAST_NAME        FIRST_NAME
 Ames             Louie
 Andriola         Leslie
.
.
.
 Lengyel          Peter
 Lonergan         Peter
57 rows selected
SQL> --
SQL> -- When the data is requested in the reverse order of the
SQL> -- ascending EMP_LAST_NAME index in the following statement,
SQL> -- the optimizer uses a reverse scan of the index to retrieve
SQL> -- the values.
SQL> SELECT LAST_NAME, FIRST_NAME FROM EMPLOYEES
  1> WHERE LAST_NAME > 'A' AND LAST_NAME < 'M'
SQL> ORDER BY LAST_NAME DESCENDING;
Conjunct        Get     Retrieval by index of relation EMPLOYEES
  Index name  EMP_LAST_NAME [1:1]        Reverse Scan
 LAST_NAME        FIRST_NAME
 Lonergan         Peter
 Lengyel          Peter
.
.
.
 Andriola         Leslie
 Ames             Louie
57 rows selected
SQL>
```

ただし、前方と後方のいずれのスキャンを使用してクエリーを処理するかをクエリーの一部として指定することはできません。例 3-50（3-120）では、オプティマイザが LAST_NAME

列に関するソート・インデックスの前方スキャンと後方スキャンがクエリーを最も迅速に処理する方法であると判断しています。他のクエリーで ORDER BY 句にソート・インデックスを指定した場合、オプティマイザはクエリーを処理するためのより効率的な方法を検出することもあります。クエリーを処理する場合にオプティマイザに特定のインデックスを使用させたいときは、クエリー・アウトラインを定義してそのインデックスを指定します。詳細は、5.9 項（5-51）を参照してください。

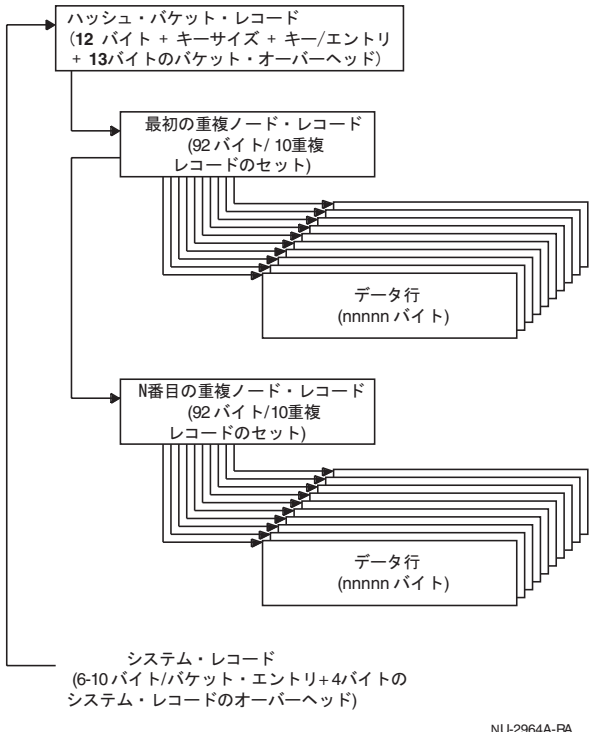
3.9.7 ハッシュ・インデックス構造

ハッシュ・インデックスは次の 3 種類のレコードで構成されます。

- ページ上の各ハッシュ・バケットへの 1 つのポインタを含むシステム・レコード。
- データ行へのポインタ、または重複を許可する場合は重複ノード・レコードへのポインタを含むハッシュ・バケット・レコード。
- 個々の重複データ行へのポインタを含む重複ノード・レコード。最大 10 個のポインタを保持し、重複データ行が多い場合は次の重複ノード・レコードへのポインタが含まれます。

例 3-11（3-122）に、ハッシュ・インデックスとデータ行が同じ記憶領域に格納された場合に、3 種類のハッシュ・インデックス・レコードをデータ行に関連付ける方法を示します。各レコードの長さをカッコ内に示します。

図 3-11 ハッシュ・インデックス構造



各レコード・タイプの詳細は、『Oracle Rdb7 Guide to Database Maintenance』を参照してください。

3.9.7.1 ハッシュ・インデックスのパフォーマンスの要因

この項では、ハッシュ・インデックスのパフォーマンスについて説明します。詳細は、8.1.3.5 項（8-33）を参照してください。

ハッシュ・インデックスを定義する場合は、次のファクタとデータベース・パラメータが重要です。

- レコード・サイズ
- ページ・サイズ
- 初期割当て
- ページ・フォーマット --- 複合ページ・フォーマットのみ

- SPAM しきい値
- SPAM 間隔
- キー・サイズ
- 一意キー値の数の見積り
- 重複行の数の見積り
- ハッシュ・インデックスを伴うテーブル内の行の総数の見積り
- ハッシュ・インデックスを格納する場所のデータに対する相対的な位置 --- 別の記憶領域か同じ記憶領域かを PLACEMENT VIA INDEX 句で指定

ハッシュ・インデックスを使用すると、データとハッシュ・インデックスが別の記憶領域に存在する場合に、2 回のみの I/O 操作で検索キーへの完全一致を検索します。また、データとハッシュ・インデックスが同じ記憶領域内で近接する場合の I/O 操作は、平均で 1 回です。サイズの計算を実行し、データ、特に重複行の分散状況を理解し、パフォーマンスがどう改善されるかを見積るのが最適な方法です。サイズの見積りの詳細は、『Oracle Rdb7 Guide to Database Design and Definition』を参照してください。

3.9.7.2 サイズ指定の問題の可能性

PLACEMENT VIA INDEX 句を使用して行を配置する場合に非常に効率的に動作するハッシング・アルゴリズムには、十分な領域を割り当てて記憶領域内のすべてのデータページに行がランダムかつ均一に分散することを保証する必要があります。多くの場合、ページ・サイズが不十分なために行がデータ・ページ内に不均一に入り始めると、ハッシュ・インデックスの効率が徐々に低下し、ついにはパフォーマンスが著しく低下することがあります。ハッシング・アルゴリズムでは検索キーから dbkey を計算するので、これは事実です。ページに空きがない場合は、記憶領域へのガイドとして設定した SPAM しきい値を使用して次の最も近い隣接ページに空き領域がないかを調べます。ターゲットとしてそのページを選択すると、これが次々に問題の原因になります。このページに空きがなくなると、シーケンス内の次のページをスキャンします。最初に十分な領域が検出されたページがその行の新しい場所になります。

やがてハッシュ・インデックスを使用して多くの行が配置されると、このハッシュ・インデックスを使用するクエリーが影響を受けるほどパフォーマンスが低下し、なんらかの処置が必要になります。検索キーに関するハッシュ検索クエリーは、dbkey を検出して行が存在するページの位置を確認します。しかし、そのページのシステム・レコードをチェックすると、ハッシュ・バケットは 10 ページ離れていることがわかります。重複ノード・レコードの dbkey は、この同じページ上にあることを示していますが、2 番目の重複ノード・レコードはさらに 60 ページ離れており、データ行はそこから 30 ページ離れているので、ずれの合計はシステム・レコードが 100 ページになり、おそらく 3 回の I/O 操作が必要になります。

この結果、重複レコードの問題とすべての重複行を処理するにはページ・サイズが小さすぎるので、期待どおりのパフォーマンスは得られません。この種の問題は、RMU Analyze コマンドを使用して検出します。

このような問題のいくつかを解決するために、3.9.7.1 項（3-122）のサマリー・リストに示した重要なパラメータの値を再計算し、次のいずれかを実行してください。

- 新しい記憶領域を定義して新しい計算値を指定します。上記の新しい記憶領域を指すように関連のストレージ・マップを変更します。SQL の ALTER STORAGE MAP 文の REORGANIZE 句を使用して新しい記憶領域にデータをロードします。未使用の空の記憶領域をすべて削除します。
- RMU Unload コマンドを使用して記憶領域からデータをアンロードします。新しい記憶領域を定義して新しい計算値を指定します。古い記憶領域を削除します。RMU Load コマンドを使用して新しい記憶領域にデータを再ロードします。
- 変更が広範囲にわたっており、そのすべてを 1 回の操作で実行する場合は、SQL の EXPORT 文と IMPORT 文を使用して必要な変更を開始します。

多くの一意キーを持つテーブルにおける検索キーへの完全一致の検索する場合に、ハッシュ・インデックスを使用するのは非常に有効でリスクの小さい方法です。ハッシュ・インデックスを使用した更新も、ソート・インデックスを使用した場合と同様に大きな利点がありますが、これらの重要なパラメータ、すなわち記憶領域の割当て、ページ・サイズ、SPAM しきい値、SPAM 間隔、ハッシュ・インデックスとデータ領域のサイズの計算に使用するパラメータの値を正確に見積らない場合は若干のリスクがあります。データベースが成長するとともに、問題を予測するには最も成長の速い部分を認識することが重要になります。

3.9.7.3 シャドウ・ページ

様々なテーブルと同じページ上のハッシュ・インデックスのクラスタ化による最適なパフォーマンスの実現が実際的でない場合、または物理的に不可能な場合は、次に優れた方法は 2 つの異なる記憶領域に関連のテーブルを格納し、シャドウ・ページを使用して最小の 2 回の I/O 操作を保証します。**シャドウイング**とは、親行から離れたもう 1 つの記憶領域に相対的に同じオフセットで対応する子行を配置することです。例 3-51（3-125）に、インデックスとストレージ・マップの定義および JOB_HISTORY 行と EMPLOYEE 行を 2 つの別々の記憶領域にシャドウイングする方法を示します。親行と両方のハッシュ・インデックスは 1 つの記憶領域に格納され、子行はもう 1 つの記憶領域に格納されます。親データ行と子データ行は、両方とも PLACEMENT VIA INDEX 句を使用して配置されます。

例 3-51 シャドウイングによる 2 つの異なる記憶領域のクラスタ化（親データ行と子データ行の格納と検索への効果）

```
CREATE UNIQUE INDEX EMPLOYEE_HASH
  ON EMPLOYEES (EMPLOYEE_ID)
  STORE IN AREA_B
  TYPE IS HASHED;
CREATE STORAGE MAP EMP_MAP
  FOR EMPLOYEES
  STORE IN AREA_B
  PLACEMENT VIA INDEX EMPLOYEE_HASH;
```

```
CREATE INDEX JOB_HISTORY_HASH
  ON JOB_HISTORY (EMPLOYEE_ID)
  STORE IN AREA_B
  TYPE IS HASHED;
CREATE STORAGE MAP JOB_HISTORY_MAP
  FOR JOB_HISTORY
  STORE IN AREA_A
  PLACEMENT VIA INDEX JOB_HISTORY_HASH;
```

シャドウイングは、EMPLOYEES テーブルのようにデータ量は少ないが JOB_HISTORY テーブルにはおそらく関連の子行がたくさんあり、一定の期間が経過するとランダムに挿入される場合には適切なストラテジです。シャドウイングによってトランザクションのグループ化が保証され、AREA_B に十分な領域を割り当てれば空き領域を使用してクラスタ化の効果を保持できます。

シャドウイングは次のように機能します。記憶領域 AREA_A の JOB_HISTORY のシャドウ行は AREA_B に格納される関連の EMPLOYEES 行と同じ相対的なオフセットでこの記憶領域に格納されていますが、ページ番号は EMPLOYEES 行と同じとは限りません。グループ化によってメモリー内バッファの使用率が向上します。記憶領域 AREA_B からバッファを読み取り、EMPLOYEES_HASH インデックスのハッシュ・インデックス・バケットを取得すると、記憶領域 AREA_B のバッファはすでにその中の EMPLOYEE 行に関連付けられています。バッファには記憶領域 AREA_A に格納された関連の JOB_HISTORY データ行へのポインタを含む JOB_HISTORY_HASH インデックスのハッシュ・バケットも含まれます。したがって、EMPLOYEES と JOB_HISTORY の両方のテーブルのハッシュ・インデックスによるアクセスで必要な I/O 操作は、AREA_B の JOB_HISTORY ハッシュ・インデックスへのアクセスが 1 回と AREA_A の JOB_HISTORY データ行へのアクセスが 1 回の合計 2 回です。AREA_A が均一な領域の場合は、行が親行と同様に分散されているので、JOB_HISTORY 行への順次アクセスのパフォーマンスも大幅に向上します。

PLACEMENT VIA INDEX 句の構文と詳細は、『Oracle Rdb7 SQL Reference Manual』を参照してください。

3.9.8 ハッシング・アルゴリズムの選択（HASHED SCATTERED か HASHED ORDERED）

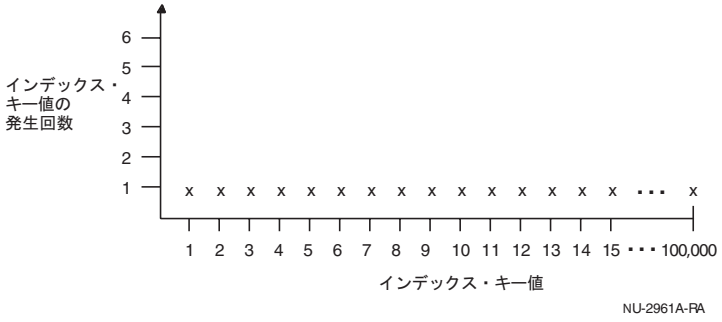
新しいハッシュ・インデックスを定義する場合は、2 つのハッシング・アルゴリズムのいずれかを選択します。Oracle Rdb は、このアルゴリズムを使用してインデックスのインデックス・キーを格納します。SQL CREATE INDEX 文で TYPE IS HASHED SCATTERED 句か TYPE IS HASHED ORDERED 句を使用します。デフォルトは SCATTERED オプションです。

HASHED SCATTERED ハッシング・アルゴリズムでは、マルチオクテット文字列がシングル・ロングワードに変換されます。さらに、それが記憶領域の初期割当てにマップされ、ターゲット・ページが決まります。このアルゴリズムの特性の 1 つは、データが領域全体に分散することです。たとえば、2 つの隣接する値が実際は記憶領域内で大きく離れている場合があります。データがソートされた順序でロードされる場合は、この分散によって高価なランダム I/O が発生するので、この場合は RMU Load コマンドの Place 修飾子か SQL INSERT 文の PLACEMENT ONLY 句を使用して、データを前処理してからロードしてください。

HASHED SCATTERED アルゴリズムでは、レコードの分散パターンは均一ではありません。一部のページは他のページよりターゲットとして選択される頻度が高くなります。一部のページは空のままなのに、他のページには複数のエントリが存在する場合もあります。

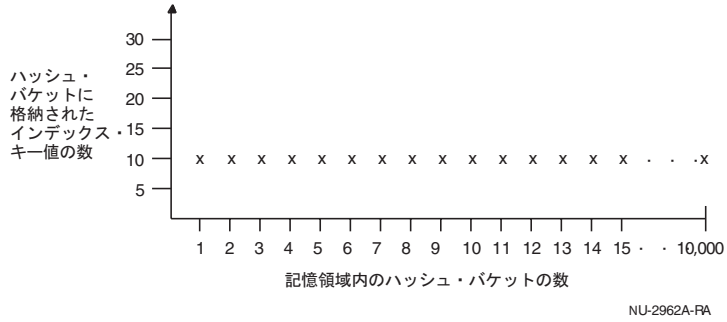
HASHED ORDERED アルゴリズムは、キー値が範囲全体に均一に分散するアプリケーションには理想的です。したがって、HASHED ORDERED アルゴリズムはアプリケーションでインデックス・キー値の範囲が指定されており、各キー値が同じ回数ずつ発生する場合に使用してください。重複値がない 1 ～ 100,000 のインデックス・キー値の範囲を使用するアプリケーションには、HASHED ORDERED アルゴリズムを使用すると効果的です。図 3-12（3-126）に、このアプリケーションで使用するインデックス・キー値の範囲のグラフを示します。アプリケーションのインデックス・キー値が図 3-12（3-126）に示すグラフ上で水平（ほぼ水平）のラインにならない場合は、HASHED ORDERED 句を指定しないでください。

図 3-12 HASHED ORDERED オプションに適したインデックス・キー値の識別



HASHED ORDERED アルゴリズムでは、先頭の 4 つのオクテットを使用して記憶領域全体に均一にデータを分散します。図 3-13（3-127）は、図 3-12（3-126）に示すインデックス・キー値が HASHED ORDERED アルゴリズムを使用して格納された結果の均一な分散を示しています。各ハッシュ・バケットには同じ数のインデックス・キーが含まれることに注意してください。インデックス・キーの格納パターンをグラフに記入すると、図 3-12（3-126）のような水平のラインが得られます。これは、インデックス・キー値が範囲全体に均一に分散する場合は、HASHED ORDERED アルゴリズムによってこれらの値を記憶領域全体に均一に分散します。

図 3-13 HASHED ORDERED オプションを適切なインデックス・キー値で使用した場合の記憶領域におけるデータの分散



HASHED ORDERED オプションは、次のすべての条件を満足する場合にのみ使用してください。

- インデックス・キーは整数値、日付、タイムスタンプ、間隔のいずれかです。

  注意: BIGINT、DATE（ANSI と VMS の両方）、TIME、TIMESTAMP、INTERVAL データ型では、ハッシュ・アルゴリズムは低次のロングワードのみを使用します。このハッシュ・アルゴリズムでは、小数の精度（位取り）は無視されます。たとえば、INTEGER(1) で指定した列に 1.1 と 2.1 の値を指定したとします。1.1 と 2.1 は 11 と 21 に同等と見なされるので、期待どおり隣接するデータベース・ページ上にはありません。

- インデックスは、ASCENDING インデックスとして定義されます。
- インデックス・キー値の範囲が存在し、各キー値の発生回数が同じです。
- MAPPING VALUES 句を使用しないでインデックスを定義しました（したがって、インデックスはすべての数値インデックス・キー値を圧縮しません）。

ハッシュ・オーダー・インデックスは、複数の列を参照する場合があります。リスト内の最後の列は、前述のデータ型の制限に従う必要があります。この列は、記憶領域内のデータの分散に使用するインデックスの唯一の要素です。しかし、リスト内の他の列はすべて STORE USING...WITH LIMIT 句で使用され、まず別々の記憶領域間にデータを分割します。

次の例に、ハッシュ・オーダー・インデックスによる複数列の参照を示します。

```
SQL> -- A company has 4 warehouses and within each warehouse are 10000
SQL> -- products.  The warehouse managers want the data for each warehouse
SQL> -- to be stored in separate storage areas with the stock rows evenly
```

```
SQL> -- distributed within each area.
SQL> --
SQL> CREATE TABLE STOCK_CONTROL
  1>   (WAREHOUSE    INTEGER,
  2>    STOCK_NO     INTEGER,
  3>    DESCRIPTION  CHAR(20) );
SQL> --
SQL> CREATE STORAGE MAP STOCK_MAP
  1>   FOR STOCK_CONTROL
  2>   STORE IN AREA_W0;
SQL> --
SQL> -- Use WAREHOUSE to partition across disks.
SQL> -- Use STOCK_NO to order the stock numbers across the area.
SQL> --
SQL> CREATE UNIQUE INDEX STOCK_INDEX
  1>   ON STOCK_CONTROL (WAREHOUSE, STOCK_NO)
  2>   TYPE IS HASHED ORDERED
  3>   STORE USING (WAREHOUSE)
  4>     IN AREA_W1 WITH LIMIT OF (1)
  5>     IN AREA_W2 WITH LIMIT OF (2)
  6>     IN AREA_W3 WITH LIMIT OF (3)
  7>     IN AREA_W4 WITH LIMIT OF (4);
SQL> COMMIT;
```

インデックス・キー値が均一に分散しており、キー値に伴うレコードを保持する記憶領域の初期サイズを決定する場合は、インデックスに対して HASHED ORDERED アルゴリズムを使用することで領域内のデータ・ページのオーバーヘッドを軽減できます。たとえば、1 ～ 100,000 の順次インデックス・キー値で重複値がない場合に、このデータを保持する記憶領域を定義するとします。これらのインデックス・キー値が含まれる 10 のレコードが 1 ページに収まることがわかっている場合は、初期割当て 10,000 データ・ページの記憶領域を定義できます（10,000 ページ×ページあたり 10 レコード =100,000 レコード）。新しい記憶領域を定義した後に HASHED SCATTERED アルゴリズムを使用してこれらのレコードを格納すると、ページあたり平均で 10 レコードが格納されますが、レコード数はページごとに変わります。したがって、記憶領域内の各ページにオーバーヘッドが必要になります。
HASHED SCATTERED アルゴリズムでは、一部のページに 11 レコード以上が格納されるためです。しかし、HASHED ORDERED アルゴリズムを使用してレコードを格納すると、各データ・ページには 10 レコード格納されます。したがって、HASHED ORDERED アルゴリズムを使用すると、各ページのオーバーヘッドを軽減できるので、記憶領域の合計サイズが削減され、ディスク領域を節約できます。

3.9.9 順次検索

Oracle Rdb がクエリーに対してどのように順次アクセス・ストラテジを選択するかは、多くの要因に依存します。1 つの要因は、レコードを選択する式（RSE）内で指定した列にインデックスが定義されているかどうかです。定義されていない場合、オプティマイザは順次検

索が最も効率的だと判断します。理由の如何にかかわらず、テーブル内の行に順次アクセスする場合は、ALG を利用できません。

順次検索では、テーブル内の各行をテストしてクエリーの RSE を満足するかどうかを判断する必要があるので、Oracle Rdb はクエリーでアクセスするすべての行のロックのレベルを上げて（昇格）行を安定に保持します。それ以外の場合は、他のトランザクションが介入して順次検索の間に行を変更する可能性があります。したがって、読取り / 書込みトランザクションの順次検索には次の特性があります。

- テーブル全体のロックが必要です。したがって、粗いロックになります。
- 保護読取りモードを使用して、現行のトランザクションを処理中に後続の更新トランザクションによって行が変更されないようにします。保護読取りモードでは、複数の更新トランザクションが同時に保護読取りリソースにアクセスすることはできません。

読取り専用トランザクションのスナップショット・ファイルの効果については、4.1.12 項（4-109）と 4.1.13 項（4-113）を参照してください。

順次検索の間は ALG が有効ではないので、Oracle Rdb はテーブル内のすべての行を他のトランザクションからアクセスできないようにする必要があります。したがって、順次検索の使用はできるだけ避けてください。例 3-52（3-129）に示すように、共有書込みモードでテーブルを予約したトランザクションがテーブルに順次アクセスする場合は、テーブルには保護書込みロックが適用されます。したがって、後続の読取り / 書込みトランザクションによるこのデータベース・リソースへのアクセスは、共有読取りトランザクションを除いてすべて拒否されます。

たとえば、インデックスを定義されていない列を使用して EMPLOYEES テーブルにアクセスするとします。オプティマイザにはインデックスを使用して直接行の位置を確認するオプションがないので、Oracle Rdb は RSE で特定された行を検出するまでテーブル自体を順次検索する必要があります。順次に処理するので、トランザクションの持続時間で検索を実行する間はテーブル全体がロックされます。

例 3-52 EMPLOYEES テーブルの順次アクセス

```
SQL> ATTACH 'FILENAME mf_personnel';
SQL> SELECT E.LAST_NAME, E.SEX, E.CITY
  1> FROM EMPLOYEES E
  2> WHERE E.POSTAL_CODE BETWEEN "03103" AND "03601"
  3> AND E.STATE = "NH";
SQL>
```

しかし、POSTAL_CODE 列と STATE 列の両方に対してインデックスを定義すると、ALG を利用できます。したがって、現行のトランザクションは RSE を満足する行とそれに関連するインデックス・ノードのみをロックします（アクセスするすべての行をロック）。ALG を使用すると、他のトランザクションはテーブル内の残りの行から選択でき、現行のトランザクションを妨害することはありません。たとえば、STATE 列に唯一のインデックスがある

場合、例 3-53（3-130）に示すクエリーは、指定した POSTAL_CODE の範囲外でも NH の値の行をすべてロックします。

例 3-53 インデックス付きアクセスと ALG

```
SQL> ATTACH 'FILENAME mf_personnel';
SQL> SELECT E.LAST_NAME, E.SEX, E.CITY
  1> FROM EMPLOYEES E
  2> WHERE E.POSTAL_CODE BETWEEN "03103" AND "03601"
  3> OR E.STATE = "NH";
SQL>
```

3.10 過剰なページ・チェックによる挿入パフォーマンス低下の確認

データベースに新しい行を格納するときにパフォーマンスが低下するという問題に直面する場合があります。これは、場合によっては Oracle Rdb がデータ・ページ上の空き領域を管理する方法が原因となります。Oracle Rdb は、行を格納する十分な領域のあるページを検出するまでに、記憶領域内の多くのページを読み取らなければならないことがあります。この項では、このような症状について説明し、前述の状況で発生し得る挿入パフォーマンスの低下を回避する方法を示します。

この項に示す状況が発生しているかどうかを確認する最も単純な方法は、Performance Monitor の「Record Statistics」表示と「File IO Overview」表示を使用することです。「Record Statistics」表示には、特定の挿入について次の 2 つのエントリが表示されます。

- pages checked
- discarded

"discarded" フィールドの値が "pages checked" に比べてかなり大きい場合は、さらに詳しく調査します。3.10.1 項（3-131）に、「File IO Overview」表示を使用して記憶領域内の過剰な I/O を特定する方法を示します。

このことを検証する別の方法は、プロセスの挿入パフォーマンスが低下した場合に Performance Monitor の「Stall Messages」画面を調べることです。その特定のプロセスに関するメッセージ "reading pages n:n to n:n" を検出します。このメッセージが消えずにページ番号が確実に増える場合は、プロセスが行を格納するのに十分な領域のあるデータベース・ページを検索している可能性があります。ただし、このストール・メッセージは Oracle Rdb がテーブルを順次読取りしている場合にも表示されます。したがって、このストール・メッセージが過剰なページ・チェックの問題を示すことを想定する前に、プロセスが挿入を行っていることを検証する必要があります。同時に、統計上の "pages checked" も着実に増大します。

Oracle Rdb は、領域管理（SPAM）ページを使用してどのデータベース・ページに新しい行を格納できる領域があるかを判断します。SPAM ページは本質的に記憶領域のページの範囲

を表すテーブルです。範囲内の各ページのテーブルにはエントリが存在し、各エントリはそれぞれのページで領域が使用可能かどうかを示します。行を格納するページを検索する場合、Oracle Rdb は各ページに十分な空き領域があるかどうかを示すエントリを検索します。ただし、SPAM ページが実際にデータ・ページの空き状況を反映するとは限らない場合があります。Oracle Rdb は、SPAM ページで空き領域があることを示すエントリを検出した場合は、このページを検索し、ページ上の実際の空き領域を調べて新しい行を格納するのに十分なスペースがあることを確認します。ページ上の空き領域が実際には行を格納するのに不十分なこともあります。このような場合を次にリストアップします。

- 複合フォーマット記憶領域または論理領域でテーブルまたはインデックスに関して設定したしきい値が不適切な場合。
- 空き領域の一部またはすべてが他のデータベース・ユーザーにロックされている場合。
- データ行のサイズを示す長さが実際のデータ行の長さを表していない場合。この状況は、常に重複を許可するインデックスの重複ノードで発生します。
- DBKEY SCOPE IS ATTACH が有効な場合。

3.10.1 記憶領域内の過剰な I/O の識別

Oracle Rdb は、行を挿入しようとするときに過剰なページをチェックするという問題を検出した場合は、「File IO Overview」画面を使用してどの記憶領域にこの動作が見られるのかを特定します。問題の原因として、空き領域のロック、不適切な SPAM しきい値、一意インデックス、レコード・サイズの見積りが考えられます。

大量のトランザクション処理アプリケーションや大量の記憶領域を扱うアプリケーションでは、「IO Statistics (by file)」画面を使ってデータベースにある記憶領域を 1 つずつチェックし、過剰な読取り操作が発生している記憶領域を特定する方法は現実的ではありません。

過剰なページ・チェックが発生している 1 つまたは複数の記憶領域を迅速に特定するには、「IO Statistics (by file)」サブメニューの「File IO Overview」画面を使用します。

「File IO Overview」画面は、同期および非同期の読取りと書込み I/O カウント、.aij ファイル、.ruj ファイル、.ace ファイル、データベース・ルート・ファイル、データ領域、スナップショット領域などすべての記憶領域についてのデータベース・ページ・チェックを表示します。

次に、「File IO Overview」画面の例を示します。

```
Node: TRIXIE         Oracle Rdb V7.0-00 Performance Monitor 19-JUN-1996 10:21:30
Rate: 0.50 Seconds      File IO Overview (Total Checked)    Elapsed: 03:39:10.39
Page: 1 of 9        KODD$:[R_ANDERSON.WORK.AIJ]OE_RDB.RDB;1         Mode: Online
-------------------------------------------------------------------------------
File/Storage.Area.Name..... Sync.Reads SyncWrites AsyncReads AsyncWrits PgCkd
All data/snap files               198752      23622      21388      57085  597k
data OE$ORDER1                       566        791       7103       2399  164k
data OE$ORDER10                      572        849       7103       2374  164k
data OE$ORDER4                       567        916       7177       2287  164k
data OE$CUSTOMER1                  15676       3060          0      11936 19096
data OE$CUSTOMER10                 15713       3158          0      11820 19096
data OE$CUSTOMER4                  15608       3633          0      11367 19096
data OE$STOCK04                     2027       4338          5        524 12136
data OE$STOCK01                     2131       3506          0       1254 11854
data OE$ITEM1                     132913        754          0       1938  6011
data OE$NEW_ORDER1                  3869        597          0       3808  5310
data OE$NEW_ORDER4                  3818        788          0       3599  5308
data OE$NEW_ORDER10                 3898        679          0       3732  5306
data OE$WAREHOUSE1                    11          3          0          8    14
data OE$WAREHOUSE10                   11          4          0          7    14
data OE$WAREHOUSE4                    11          5          0          6    14
ACE (AIJ Cache Electronic)             0          0          0          0     0
-------------------------------------------------------------------------------
Config Exit Help Menu >next_page <prev_page Options Reset Set_rate Unreset Write
```

この「File IO Overview」画面では、同期および非同期の読取りおよび書込みカウントが表示され、ページ・チェックの合計で、降順にソートされています。データベース記憶領域には、接頭辞 data 、スナップショット領域には、接頭辞 snap が表示されます。データベースに追加した記憶領域は、自動的に画面表示されます。

"PgCkd" 行は、各領域でチェックしたデータベース・ページの数であり、合計値が "all data/snap files" に表示されます。

「File IO Overview」画面の設定により、記憶領域情報をいくつかの順序でソートできます。[C] を押すと、画面の水平メニューから「Config」オプションが選択され、次のような構成メニューが表示されます。

```
+------ Select Display Configuration -----+
|                                          |
|  A.  Unsorted totals display             |
|  B.  Sort by total synchronous reads     |
|  C.  Sort by total synchronous writes    |
|  D.  Sort by total asynchronous reads    |
|  E.  Sort by total asynchronous writes   |
|  F.  Sort by total reads & writes        |
|  G.  Sort by total pages checked         |
|  H.  Unsorted current rate display       |
|  I.  Sort by synchronous read rate       |
|  J.  Sort by synchronous write rate      |
|  K.  Sort by asynchronous read rate      |
|  L.  Sort by asynchronous write rate     |
|  M.  Sort by total current I/O rates     |
|  N.  Sort by pages checked rates         |
+------------------------------------------+
```

オプション「B」から「N」までを選択すると、2 つのソート方法を持つ記憶領域がアルファベット順に表示されます。

「File IO Overview」画面のヘッダーに表示されている名前は、選択した構成を示しています。

INPUT コマンド行修飾子を使って出力データ・ファイルを再生する場合、「File IO Overview」画面は使用できません。

構成オプションの詳細は、Performance Monitor のヘルプを参照してください。

3.10.2 不適切なしきい値の設定

次のいずれかの状態で、行をテーブルに格納するときに過剰なページ・チェックが発生すると仮定します。

- テーブルを複合フォーマット領域に格納。
- テーブルには、領域マップで定義された論理領域のしきい値がある。
- テーブルには、論理領域しきい値を含むインデックスが定義されている。

このような状態で過剰なページ・チェックが発生する場合、不適切なしきい値の設定が原因となっている可能性があります。しきい値の定義で使用した計算式を検証し、しきい値が正しく設定されていることを確認してください。

3.10.3 空き領域のロック

プロセスがデータベースから行を削除して、データベースへの接続の間に dbkey スコープを設定すると、削除した行に割り当てられていた領域は、プロセスがデータベースから切断するまでは、予約済みつまりロックされた状態になります。トランザクションがロールバック

された場合に、行をページへ確実に戻すため、このような処理を行います。行を削除すると、SPAM ページを更新し、削除によって使用可能になった領域を反映します。他のデータベース・プロセスが、行の格納に使用できる領域を検索しようと SPAM ページを参照すると、SPAM ページは、行を削除したページに使用可能な領域があることを示します。しかし、実際にページを取り出すと、ページ上にある使用可能な領域の一部または全部は他のプロセスによってロックされており、行を格納できるだけの領域がないことがわかります。したがって、行を格納するために他のページを選択する必要があり、同じ処理を繰り返すことになります。

一般的には、このような処理がデータベース・パフォーマンスに影響をおよぼすことはほとんどありませんが、多数のページ上で行を削除するデータベース・メンテナンスを長時間アクティブな状態にしておく場合などは、ロックされた空き領域を含むデータベース・ページが数多く存在することになります。メンテナンスの実行中に、これと同じ領域に行を格納しようとするプロセスが多数存在すると、パフォーマンスに大きな影響を与えます。

3.10.4 AIP に格納された長さはテーブル行の実際の行さを反映

Oracle Rdb は、テーブル行の通常の長さをエリア・インベントリ・ページ（Area Inventory Page：AIP）と呼ばれる構造に格納します。AIP に格納されている長さによって、均一フォーマット領域のページがいっぱいになるまでにはどの程度の領域が使用可能かを判断します。次のような場合、AIP に格納されている行の長さは、実際の行長ではありません 。

- 重複を許可するインデックス
- 変更され、行を追加または削除した一意のインデックスやテーブル
- セグメント文字列（LIST OF BYTE VARYING データ型）

3.10.4.1 重複を許可するインデックスでの AIP 長の問題

インデックスが重複を許可する場合、AIP に格納されている長さは、実際のインデックス・ノード・サイズに関係なく、215 バイトになります。インデックスが重複を許可する場合、ノード・サイズが可変である可能性があるので、インデックスの論理領域にある行の長さには、215 バイトを平均値として使用します。AIP の行サイズが実際の行長よりも小さい場合、フルサイズの行を追加で格納できないとしても、SPAM エントリは、ページに使用可能領域があるとみなします。これが、挿入パフォーマンスの問題が発生する一般的な原因です。

たとえば、インデックス・ノード・サイズとして 430 バイト（一般的なデフォルト値）を使用し、インデックスが格納されている記憶領域のページ・サイズが 2 ブロックだとします。ページ・オーバーヘッドを差し引くと、2 ブロック・ページ上で使用可能な領域は 982 バイトです。この例では、最初にページは空だとします。

1. フルサイズ（430 バイト）のインデックス・ノードを格納します。ページに格納されている各行には、8 バイトのオーバーヘッドがあるので、ページに残っている空きバイト数は 982 - 430 - 8 = 544 となります。

2. インデックス・ノードで重複キー・エントリが実行されたため、重複ノードが同じページ上に作成されます。最初の重複ノードは 112 バイトです（重複ノードのサイズは、作成されたときによって異なりますが、この例では 112 バイトを使用します）。ページ上に残っている空きバイト数は、544 - 112 - 8 = 424 バイトとなります。

この時点では、ページには 424 バイト残っています。この値は、論理領域の行長として AIP が示す 215 バイトよりも大きいため、SPAM ページは、ページに使用可能な領域があると表示します。しかし、残りの空き領域（424 バイト）は、430 バイト・ノードの格納には足りないため、ページ上でフルサイズのインデックス・ノードを格納しようとすると失敗します。

したがって、他のページを SPAM ページで選択する必要があり、実際に十分な空き領域が見つかるまでプロセスはこの作業を続けます。論理領域に重複ノードが数多く存在する場合、論理領域のページは、かなりの部分がこの状態になっています。フルサイズのインデックス・ノードを新しく格納するときに、行の格納に使用できるページが見つかるまで、ページの読取りやチェックを何回も実行しなければなりません。

この問題は、論理領域しきい値を使っても解決できません。フルサイズのインデックス・ノードを格納するには領域が不十分な場合に、ページがいっぱいであることを SPAM ページで示すようなしきい値を設定する必要があります。

次に、同じ例を使って、2 ブロック・ページにノード・サイズ 430 バイトのインデックスで過剰なページ・チェックが発生しないように、論理領域のしきい値を正しく設定する方法を説明します。適正なしきい値を計算するには、まず最初に、フルサイズのノードをページに格納できなくなった時点で、ページがどの程度使用済みになっているかを把握する必要があります。この例では、データベース・ページが領域不足の状態になる直前の最大使用済み容量は、982 - 430 - 8 = 544 バイトとなります。したがって、使用済みサイズが 544 バイト以下ならば、フルサイズ・ノードをもう 1 つ格納できます。しきい値は、544 / 982 = .553971、つまり 55% となります。

次に、上記の内容のインデックスを作成する例を示します。

```
SQL> CREATE INDEX TEST_INDEX ON EMPLOYEES (LAST_NAME)
  1>     STORE IN RDB$SYSTEM
  2>     (THRESHOLD IS (55));
```

圧縮を有効にした通常のテーブルで使用する圧縮アルゴリズムは、インデックス・ノードには適用しない点に注意してください。インデックス・ノードは、データ行のように圧縮されることはなく、ノード・サイズに指定したバイト数を常に使用します。インデックスのしきい値を計算するときは、圧縮を考慮しないでください。

3.10.4.2 セグメント文字列での AIP 長の問題

AIP に格納されているセグメント文字列の長さは、実際の長さにかかわらず常に 150 バイトです。したがって、3.10.4.1 項（3-134）で説明した同じ問題が、セグメント文字列でも発生します。均一領域内のページに 158（150 + 8 オーバーヘッド・バイト = 158）バイト以上の空き領域がある場合、このページの SPAM エントリは、使用可能な領域があることを示し

ます。しかし、実際の空き領域よりも大きなセグメント文字列を格納しようとすると、セグメントを格納するだけの空き領域がないために、ページはチェックされます。

この状態を避けるには、複合フォーマットの記憶領域のみにセグメント文字列を格納してください。記憶領域には、格納するセグメント・サイズに適切なしきい値を定義しておいてください。

3.10.5 DBKEY SCOPE IS ATTACH 句の使用

DBKEY SCOPE IS ATTACH を使ってデータベースを宣言すると、Oracle Rdb は、ページを使用する前に SPAM しきい値を再計算し、十分な空き領域があることを確認します。したがって、SPAM エントリが、空き領域が存在することを示している場合、Oracle Rdb はページをフェッチして、ページの使用済み領域を再計算します。Oracle Rdb による SPAM しきい値の再計算では、DBKEY SCOPE IS ATTACH 句を使用することによって発生するオーバーヘッドが考慮されています。Oracle Rdb は、ページの使用済み領域を再計算した後、フルサイズ・レコードを格納できないと判断すると（追加のオーバーヘッドが原因）、実際のセグメントが格納可能かどうかに関係なく、ページは使用済みであるとみなします。これにより、過剰なページ・チェックが発生する可能性があります。

4

パラメータの調整

この章では、データベース・パフォーマンスの向上を目的に、調整可能なデータベース・パラメータと OpenVMS システム・パラメータについて説明します。ここでは、グローバル・バッファなどの Oracle Rdb 機能を有効にする判断基準と、有効にした場合のパラメータの推奨値について、ガイドラインを示します。

4.1 データベース・パラメータの調整

表 4-1（4-2）は、アプリケーションの物理設計に必要な内容を、データベースの種類（簡易または複雑）とデータベースのサイズ（小規模から非常に大規模）ごとに示しています。Oracle Rdb のデフォルト値の変更を検討する前に、どの程度までデフォルト値を使用できるのか示しています。

表 4-1 Oracle Rdb のデフォルト値の適用範囲

	チューニングの程度	
データベース・サイズ	**簡易データベース テーブルは 10 ～ 15 未満**	**複雑なデータベース テーブルは 15 以上**
10Mb 未満	なし	最小限
10 ～ 100Mb	最小限	中程度
100 Mb ～ 1 Gb 超	中程度	大幅な変更
1 Gb 超	大幅な変更	大幅な変更

チューニング作業

- **なし** --- 事実上、チューニングは実施しない。単一ファイルのデータベース、均一のページ・フォーマット、ソート済みインデックスなど。デフォルト値は、すべてでなくともほとんどを採用。
- **最小限** --- 最小限のチューニングを実施。単一ファイルまたは複数ファイル、インデックスの適切な定義、バッファの適切な設定など。一部のデフォルト値を変更。
- **中程度** --- 中程度のチューニングを実施。複数ファイルのデータベース。複合ページ・フォーマット、ハッシュ・インデックスの使用、複合ページ・フォーマットでのセグメント文字列の使用、SPAM パラメータの設定。デフォルト値の多くを変更。
- **大幅な変更** --- 大幅なチューニングを実施。複数ファイル・データベース。すべての機能の組合せや定義を詳細にチェック。すべてではなくとも、デフォルト値を変更。ハッシュ・インデックス、B ツリー・ノードのサイジング。

表 4-2（4-2）は、Oracle Rdb において、データベース全体のパラメータに設定されているデフォルト値です。

表 4-2 Oracle Rdb のデータベース全体のパラメータ値デフォルト、最小、最大

データベース・ワイド・パラメータ	デフォルト値 （最小 / 最大）
説明	なし
パス名	CDD$DEFAULT ◆

OpenVMS VAX OpenVMS Alpha

表 4-2 Oracle Rdb のデータベース全体のパラメータ値デフォルト、最小、最大（続き）

データベース・ワイド・パラメータ	デフォルト値 （最小 / 最大）
ユーザー数	50 (1/2032)
バッファ数	20 (2/32768)
クラスタ・ノードの最大数	16 (1/96)
リカバリ・バッファの数	20 (2/32768)
バッファ・サイズ	最大領域ページ・サイズの 3 倍
グローバル・バッファ	無効
行キャッシング	無効
高速コミット ジャーナルの最適化	無効 無効
キャリーオーバー・ロックの最適化	有効
ロックの粒度	有効
スナップショット・ファイルの有効 / 無効	有効
即時 / 遅延スナップショット	即時
データベース・オープン	オープンは自動
After-image ジャーナルの指定	AIJ は名前を指定するまで無効
After-image ジャーナルの割当て	512 ブロック（0/ ディスク・デバイス・サイズ）
After-image ジャーナルの拡張	512 ブロック（拡張オプションなし） 512 ブロック（0/ ディスク・デバイス・サイズ）
ディクショナリの必要性	ディクショナリは不要 ◆
ディクショナリの使用	ディクショナリを使用 ◆

OpenVMS VAX OpenVMS Alpha

OpenVMS VAX OpenVMS Alpha

次の項では、データベース・パラメータについて説明するので、データベース・アプリケーション向けに最適な値を決定する参考にしてください。Oracle Rdb は、データベース・パラメータをデフォルト値に設定しても十分なパフォーマンスを発揮できるので、特殊なアプリケーション向けにデータベースを調整するときのガイドラインとして活用してください。

アプリケーション環境の中で、データベース・パフォーマンスに影響を与える要因には、使用可能な仮想メモリー、ユーザー数、読取り専用や更新集中型などのアプリケーションの特徴があります。4.1.1 項（4-7）で説明した「Performance Monitor」画面を使用すると、データベース・パフォーマンスやデータベース・ファイルに格納されているデータ領域を監視で

きるので、データベース・パラメータの調整が必要かどうかを判断してください。変更は、SQL で実行できます。

データベースの作成時に、データベースの定義として、Oracle Rdb のデフォルト・パラメータをすべて使用するか、各サイトのニーズに合わせて一部パラメータを変更するかを選択します。ほとんどの場合、特にデータベース・アプリケーションの開発フェーズなどでは、Oracle Rdb は、デフォルト設定で 十分なパフォーマンスを発揮します。表 4-1（4-2）は、Oracle Rdb のデフォルト値を適用できるかどうかを、データベース・サイズ（行数）とデータベースの複雑さ（テーブル数）ごとに示しています。データベース・パラメータを変更するには、SQL ALTER DATABASE または IMPORT 文に、データベース・パラメータの新しい値を指定して実行します。『Oracle Rdb7 Guide to Database Design and Definition』では、ユーザーがデータベースに接続した状態で更新可能なパラメータや、ユーザーがデータベースに接続していない状態でのみ更新可能なパラメータなど、データベースと記憶領域の変更について説明しています。

表 4-3（4-4）は、データベース・ワイド・パラメータの指定が可能な SQL 文を示しています。

表 4-3 データベース・パラメータを変更する SQL 文

データベース・ワイド・パラメータ	SQL CREATE DATABASE	SQL ALTER DATABASE	SQL IMPORT DATABASE
説明	可	不可	可
パス名	可	不可	可 ◆
照合順序は	可	不可	可
ユーザー数	可	可（複数 のみ）	可
バッファ数	可	可	可
クラスタ・ノードの最大数	可	可（複数 のみ）	可
リカバリ・バッファの数	可	可	可
バッファ・サイズ	可	可	可
グローバル・バッファの有効 / 無効	可	可	不可
行キャッシュの有効 / 無効	可	可	不可
高速コミットの有効 / 無効	不可	可	不可
ジャーナルの最適化	不可	可	不可
キャリーオーバー・ロックの最適化	可	可	不可
ロックの粒度	可	可	可

表 4-3 データベース・パラメータを変更する SQL 文（続き）

データベース・ワイド・パラメータ	SQL CREATE DATABASE	SQL ALTER DATABASE	SQL IMPORT DATABASE
スナップショット・ファイルの有効 / 無効	可	可	可
即時 / 遅延スナップショット	可	可	可
データベース・オープン	不可	可	不可
After-image ジャーナルの指定	不可	可	不可
After-image ジャーナルを指定しない	不可（デフォルト）	可	不可（デフォルト）
ジャーナルの割当て	不可	可	不可
ジャーナルの拡張	不可	可	不可
スナップショット・ファイルの割当て	可	可	可
スナップショット・ファイルの拡張	可	可	可
ディクショナリの必要性	可	可	可 ◆
ディクショナリの使用	可	不可	可 ◆
セグメント文字列記憶領域	可	不可	可

表 4-4 (4-5) は、データベース・ワイド・パラメータを指定する Oracle RMU コマンドを示しています。

表 4-4 データベース・パラメータを変更する Oracle RMU コマンド

データベース・ワイド・パラメータ	RMU Restore	RMU Copy_Database	RMU Move_Area
説明	不可	不可	不可
パス名	可	不可	不可 ◆
照合順序は	不可	不可	不可
ユーザー数	可	可	可
バッファ数	可	不可	不可
クラスタ・ノードの最大数	可	可	可
リカバリ・バッファの数	不可	不可	不可

表 4-4 データベース・パラメータを変更する Oracle RMU コマンド（続き）

データベース・ワイド パラメータ	RMU Restore	RMU Copy_Database	RMU Move_Area
バッファ・サイズ	可	不可	不可
行キャッシュ・サイズ	不可	不可	不可
グローバル・バッファの有効 / 無効	可	不可	不可
高速コミットの有効 / 無効	不可	不可	不可
ジャーナルの最適化	不可	不可	不可
キャリーオーバー・ロック最適化の有効 / 無効	不可	不可	不可
ロックの粒度	不可	不可	不可
スナップショット・ファイルの有効 / 無効	不可	不可	不可
即時 / 遅延スナップショット	不可	不可	不可
データベース・オープン	可	不可	不可
After-image ジャーナルの指定	可	可	可
After-image ジャーナルを指定しない	可	可	可
ジャーナルの割当て	可	可	不可
ジャーナルの拡張	可	可	不可
スナップショット・ファイルの割当て	不可	不可	不可
スナップショット・ファイルの拡張	不可	不可	不可
OpenVMS VAX OpenVMS Alpha ディクショナリの必要性	不可	不可	不可 ◆
OpenVMS VAX OpenVMS Alpha ディクショナリの使用	可	不可	不可 ◆
セグメント文字列記憶領域	不可	不可	不可

RMU Restore、RMU Copy_Database、RMU Move_Area コマンドを使用したデータベース・パラメータの変更方法については、『Oracle Rdb7 Guide to Database Maintenance』および『Oracle RMU Reference Manual』を参照してください。

4.1.1 データベース・パラメータ情報の収集

4.1.1.1 項（4-7）から 4.1.1.12 項（4-18）では、Performance Monitor の各画面を使用したデータベース・パラメータのモニターについて説明します。

4.1.1.1 Performance Monitor の「Database Parameter Information」サブメニュー

Performance Monitor では、一連の画面で動的な情報を表示します。データベース・パラメータを変更すると、画面上の情報は自動的に更新されます。メイン・メニューから「Database Parameter Information」オプションを選択すると、次のサブメニューが表示されます。

```
Node: TRIXIE          Oracle Rdb V7.0-00 Performance Monitor 28-MAY-1996 14:19:16
Rate: 3.00 Seconds            Summary IO Statistics          Elapsed: 03:58:22.56
Page: 1 of 1        RDBVMS_USER1:[LOGAN.V70]MF_PERSONNEL.RDB;1       Mode: Online
-------------------   +--------------------------------+
statistic.........    |         Select Display         |..... average......
name..............    |                                |..... per.trans....
transactions          |  A. General Information        |      7          1.0
verb successes        |  B. Buffer Information         |   4809        687.0
verb failures         |  C. Lock Information           |    241         34.4
                      |  D. Storage Area Information   |
synch data reads      |  E. Row Cache Information      |    259         37.0
synch data writes     |  F. Journaling Information     |      0          0.0
asynch data reads     |  G. Journal Information        |    105         15.0
asynch data writes    |  H. Fast Commit Information    |      0          0.0
RUJ file reads        |  I. Hot Standby Information    |      0          0.0
RUJ file writes       |  J. Audit Information          |      5          0.7
AIJ file reads        |  K. Active User Information    |      5          0.7
AIJ file writes       |  L. OpenVMS SYSGEN Parameters  |      0          0.0
ACE file reads        |                                |      0          0.0
ACE file writes       +--------------------------------+
root file reads                                           64          9.1
root file writes              0        0        0.0       31          4.4
-------------------------------------------------------------------------------
Type <return> or <letter> to select option, <1rol-Z> to cancel
```

次に、「Buffer Information」画面の例を示します。

```
Node: TRIXIE         Oracle Rdb V7.0-00 Performance Monitor 28-MAY-1996 14:29:15
Rate: 3.00 Seconds             Buffer Information           Elapsed: 04:08:21.21
Page: 1 of 1       RDBVMS_USER1:[LOGAN.V70]MF_PERSONNEL.RDB;1       Mode: Online
-------------------------------------------------------------------------------
Default user buffer count is 20
Default recovery buffer count is 20
Buffer size is 6 blocks
Global Buffers are enabled
- Global buffer count is 250
- Maximum global buffer count per user is 5
- Page transfer via memory is disabled
Global section size with global buffers disabled is 610007 bytes
- With global buffers enabled is 1522877 bytes
Asynchronous pre-fetch is enabled
- Maximum pre-fetch depth is 5 buffers
Detected asynchronous pre-fetch is enabled
- Maximum pre-fetch depth is 4 buffers
- Pre-fetch threshold is 4 pages
Asynchronous batch-write is enabled
- Clean buffer count is 5
- Maximum batch size is 4 buffers
-------------------------------------------------------------------------------
Exit Help Menu Options Refresh Set_rate Write !
```

「Database Parameter Information」画面は、OUTPUT 修飾子を指定して作成したバイナリ出力ファイルには記録されません。したがって、INPUT 修飾子を指定してバイナリ・ファイルを再生しても、この画面は参照できません。

画面の詳細は、Performance Monitor のヘルプを参照してください。

4.1.1.2 Performance Monitor の「PIO Statistics--Data Writes」画面

「PIO Statistics--Data Writes」画面は、データ・ファイル書込みとバッファのマーク解除アクティビティ（ディスクへのバッファ書込み）に関する情報を表示します。次に、「PIO Statistics--Data Writes」画面の例を示します。

```
Node: TRIXIE          Oracle Rdb V7.0-00 Performance Monitor 17-MAY-1996 16:30:38
Rate: 3.00 Seconds         PIO Statistics--Data Writes       Elapsed: 00:00:39.01
Page: 1 of 1       RDBVMS_USER1:[LOGAN.V70]MF_PERSONNEL.RDB;1       Mode: Online
-------------------------------------------------------------------------------
statistic.........      rate.per.second.............. total....... average......
name..............      max..... cur..... avg....... count....... per.trans....
unmark buffer                22        0        9.7        12043          76.7
  transaction                 0        0        0.1           73           0.5
  pool overflow              22        0        9.4        11706          74.6
  blocking AST                0        0        0.0            0           0.0
  lock quota                  0        0        0.0            0           0.0
  lock conflict               0        0        0.1          165           1.1
  user unbind                 1        0        0.0           22           0.1
  batch rollback              0        0        0.0            0           0.0
  new area mode               0        0        0.0            0           0.0
  larea change                0        0        0.0           56           0.4
  incr backup                 0        0        0.0            1           0.0
  no AIJ access               0        0        0.0            0           0.0
  truncate snaps              0        0        0.0            0           0.0
  checkpoint                  0        0        0.0           20           0.1
  AIJ backup                  0        0        0.0            0           0.0
-------------------------------------------------------------------------------
Exit Graph Help Menu Options Pause Reset Set_rate Time_plot Unreset Write X_plot
```

画面の各フィールドについては、Performance Monitor のヘルプを参照してください。

4.1.1.3 Performance Monitor の「PIO Statistics--Data Fetches」画面

次に、ローカル・バッファが有効化されたデータベースに関する「PIO Statistics-Data Fetches」画面の例を示します。「PIO Statistics--Data Fetches」画面は、データ・ページ要求の処理に関する統計を表示します。この画面には、ローカル・バッファが有効化されているデータベース用のフォーマットと、グローバル・バッファが有効化されているデータベース用のフォーマットの 2 つがあります。

```
Node: TRIXIE         Oracle Rdb V7.0-00 Performance Monitor 24-MAY-1996 09:45:46
Rate: 3.00 Seconds        PIO Statistics--Data Fetches      Elapsed: 00:05:59.16
Page: 1 of 1      RDBVMS_USER1:[LOGAN.LOCAL]MF_PERSONNEL.RDB;1      Mode: Online
-------------------------------------------------------------------------------
statistic.........      rate.per.second.............. total....... average......
name..............      max..... cur..... avg....... count....... per.trans....
fetch for read              2287        0       20.4        18524         926.2
fetch for write               58        0        0.4          348          17.4
in LB: all ok               2035        0       18.3        16631         831.6
   LB: need lock             296        0        2.1         1949          97.5
   LB: old version             0        0        0.0            0           0.0
not found: read               34        0        0.3          292          14.6
         : synth               0        0        0.0            0           0.0
DAPF: success                  0        0        0.0            0           0.0
DAPF: failure                  0        0        0.0            0           0.0
DAPF: utilized                 0        0        0.0            0           0.0
DAPF: discarded                0        0        0.0            0           0.0
-------------------------------------------------------------------------------
Exit Graph Help Menu Options Pause Reset Set_rate Time_plot Unreset Write X_plot
```

画面の各フィールドについては、Performance Monitor のヘルプを参照してください。

4.1.1.4 Performance Monitor の「PIO Statistics--SPAM Fetches」画面

「PIO Statistics--SPAM Fetches」画面は、SPAM ページ要求の処理に関する統計が表示されます。この画面には、ローカル・バッファが有効化されているデータベース用のフォーマットと、グローバル・バッファが有効化されているデータベース用のフォーマットの 2 つがあります。次に、グローバル・バッファが有効化されたデータベースに関する「PIO Statistics--SPAM Fetches」画面の例を示します。

```
Node: TRIXIE          Oracle Rdb V7.0-00 Performance Monitor 24-MAY-1996 10:02:33
Rate: 3.00 Seconds         PIO Statistics--SPAM Fetches      Elapsed: 00:00:24.49
Page: 1 of 1    RDBVMS_USER1:[LOGAN.GLOBAL]MF_PERSONNEL.RDB;1       Mode: Online
-------------------------------------------------------------------------------
statistic.........      rate.per.second.............. total....... average......
name..............      max..... cur..... avg....... count....... per.trans....
fetch for read                5        0        0.1           59           3.0
fetch for write               0        0        0.0            0           0.0
in AS: all ok                 4        0        0.0           52           2.6
   AS: lock for GB            0        0        0.0            0           0.0
   AS: need lock              0        0        0.0            0           0.0
   AS: old version            0        0        0.0            0           0.0
in GB: need lock              1        0        0.0            6           0.3
   GB: old version            0        0        0.0            0           0.0
   GB: transferred            0        0        0.0            0           0.0
not found: read               0        0        0.0            1           0.1
         : synth              0        0        0.0            0           0.0
DAPF: success                 0        0        0.0            0           0.0
DAPF: failure                 0        0        0.0            0           0.0
DAPF: utilized                0        0        0.0            0           0.0
DAPF: discarded               0        0        0.0            0           0.0
-------------------------------------------------------------------------------
Exit Graph Help Menu Options Pause Reset Set_rate Time_plot Unreset Write X_plot
```

画面の各フィールドについては、Performance Monitor のヘルプを参照してください。

4.1.1.5 Performance Monitor の「Asynchronous IO Statistics」画面

「Asynchronous IO Statistics」画面は、データベース・ファイルへの非同期の読取りと書込みに関する情報を表示します。ストール時間が 1/100 秒単位で表示されます。

この画面では、非同期読取りおよび書込み操作が要求された回数と、実際に実行した非同期読取りおよび書込みの数が表示されます。

「Asynchronous IO Statistics」画面は、「IO Statistics」サブメニューから選択できます。次に、「Asynchronous IO Statistics」画面の例を示します。

```
Node: TRIXIE          Oracle Rdb V7.0-00 Performance Monitor 24-MAY-1996 10:13:22
Rate: 3.00 Seconds          Asynchronous IO Statistics       Elapsed: 00:11:13.90
Page: 1 of 1          SQL_DISK1:[USER]MF_PERSONNEL.RDB;1            Mode: Online
-------------------------------------------------------------------------------
statistic.........      rate.per.second.............. total....... average......
name..............      max..... cur..... avg....... count....... per.trans....
data read request             67        0        6.3         2976        1488.0
data read IO                  14        0        2.1          988         494.0
spam read request             67        0        2.5         1161         580.5
spam read IO                   0        0        0.0            4           2.0
read stall count               1        0        0.1           51          25.5
read stall time                4        0        0.3          153          76.5
write IO                       8        0        1.7          812         406.0
write stall count              3        0        0.1           51          25.5
write stall time              24        0        0.5          223         111.5
-------------------------------------------------------------------------------
Exit Graph Help Menu Options Pause Reset Set_rate Time_plot Unreset Write X_plot
```

画面の各フィールドについては、Performance Monitor のヘルプを参照してください。

4.1.1.6 Performance Monitor の「Process Accounting」画面

OpenVMS VAX OpenVMS Alpha

「Process Accounting」画面は、常に更新されているローカル・プロセスのアカウンティング情報を表示します。この画面は、OpenVMS SHOW PROCESS/CONTINUOUS ユーティリティのかわりに提供されています。この画面は、同じ内容を表示しますが、指定したノード上にあるすべてのアクティブなデータベース・プロセスを同時にモニターできる点が異なります。

この画面では、オペレーティング・システムのアカウンティング情報を直接表示するので、データベース管理者は、この画面を使って、データベース・プロセスが使用しているシステム・リソースを評価できます。「Process Accounting」画面に表示されている値は、データベース内で発生しているアクティビティだけでなく、すべてのプロセス・アクティビティを示します。したがって、アプリケーション処理全体をモニターしたいときに便利です。

「Process Accounting」画面は、「Process Information」サブメニューから選択できます。

この画面は、動的に変化する情報のみを表示します。したがって、各プロセスで固定されているクオータなどの情報は、他の情報で取得できるため、この画面では表示されません。「Process Accounting」画面には、Brief モードと Full モードがあり、モードによってアクティブなデータベース・プロセスに関する表示内容が変わります。

[B] を押すと、Brief モードを選択できます。Brief モードでは、プロセス ID、プロセス名、CPU 時間、残りのロック・クオータ・カウント、ページ・フォルト・カウント、ダイレクト I/O 操作の数、ワーキング・セット・サイズ、仮想メモリー・サイズなどが、各プロセス 1 行で表示されます。次に、Brief モードでの「Process Accounting」画面の例を示します。

```
Node: TRIXIE         Oracle Rdb V7.0-00 Performance Monitor 24-MAY-1996 10:22:20
Rate: 3.00 Seconds             Process Accounting           Elapsed: 00:20:11.30
Page: 1 of 1         SQL_DISK1:[RICK.V70]MF_PERSONNEL.RDB;1         Mode: Online
------------------------------------------------------------------------------
Process.ID Process.name... CPUtime.... EnqCnt. PGflts. NumDio. WSsize. VMsize.
6E2012D3:1 RICK            00:00:15.46   17809   20775     808    4471   15082
6E201887:1 SMITH           00:00:11.23   17882   17383     444    3606   14426
------------------------------------------------------------------------------
Exit Full Help Menu >next_page <prev_page Set_rate Write Zoom !
```

[F] を押すと、Full モードを選択できます。Full モードでは、ユーザー名、イメージ名、プロセス・ステート、ページ・ファイル・クオータ・カウント、ダイレクト I/O クオータ・カウント、バッファ I/O 操作の数、バッファ I/O クオータ・カウントなどが、2 行目に表示されます。次に、Full モードでの「Process Accounting」画面の例を示します。

```
Node: TRIXIE         Oracle Rdb V7.0-00 Performance Monitor 24-MAY-1996 10:24:48
Rate: 3.00 Seconds             Process Accounting           Elapsed: 00:22:39.16
Page: 1 of 1         SQL_DISK1:[RICK.V70]MF_PERSONNEL.RDB;1         Mode: Online
------------------------------------------------------------------------------
Process.ID Process.name... CPUtime.... EnqCnt. PGflts. NumDio. WSsize. VMsize.
User.name... Image.name.............. State.. PGfCnt. DioCnt. NumBio. BioCnt.
6E2012D3:1 RICK            00:00:15.55   17796   20776     808    4472   15082
RICK         SQL$601                     LEF      124448     200    1464     199
6E201887:1 SMITH           00:00:11.92   17858   18378     516    5000   14954
SMITH        SQL$601                     LEF      124576     200     515     199
------------------------------------------------------------------------------
Brief Exit Help Menu >next_page <prev_page Set_rate Write Zoom !
```

「Process Accounting」画面はリセットできないので注意してください。情報はリアルタイムに収集されるので、出力ファイルへ書き込むことはできません（画面では OUTPUT 修飾子が使用できません）。したがって、入力ファイルの再生では表示できません。

「Process Accounting」画面の各フィールドについては、Performance Monitor のヘルプを参照してください。◆

4.1.1.7 Performance Monitor の「Record Statistics」画面

「Record Statistics」画面は、データベースの記憶領域のデータ行アクティビティ（マーク、フェッチ、格納、削除した行）の概要を表示します。次に、「Record Statistics」画面の例を示します。

```
Node: TRIXIE         Oracle Rdb V7.0-00 Performance Monitor 24-MAY-1996 13:21:49
Rate: 3.00 Seconds             Record Statistics            Elapsed: 03:19:40.61
Page: 1 of 1     KODD$:[R_ANDERSON.WORK.AIJ]MF_PERSONNEL.RDB;1      Mode: Online
-------------------------------------------------------------------------------
statistic.........      rate.per.second.............. total....... average......
name..............      max..... cur..... avg....... count....... per.trans....
record marked                12       11        4.7         3784           2.1
record fetched               41       36       16.0        12730           7.1
    fragmented                0        0        0.0            0           0.0
  record stored               6        5        2.3         1815           1.0
    fragmented                0        0        0.0            0           0.0
  pages checked               6        5        2.3         1815           1.0
      saved IO                0        0        0.0            0           0.0
     discarded                0        0        0.0            0           0.0
record erased                 0        0        0.0            0           0.0
   fragmented                 0        0        0.0            0           0.0
-------------------------------------------------------------------------------
Exit Graph Help Menu Options Pause Reset Set_rate Time_plot Unreset Write X_plot
```

画面の各フィールドについては、Performance Monitor のヘルプを参照してください。

4.1.1.8 Performance Monitor の「AIJ Statistics」画面

「AIJ Statistics」画面は、After-image ジャーナルの論理的および物理的なアクティビティをモニターします。次に、「AIJ Statistics」画面の例を示します。

```
Node: TRIXIE         Oracle Rdb V7.0-00 Performance Monitor 24-MAY-1996 13:29:50
Rate: 3.00 Seconds              AIJ Statistics              Elapsed: 03:27:41.92
Page: 1 of 1         SQL_DISK1:[RICK.V70]MF_PERSONNEL.RDB;1         Mode: Online
-------------------------------------------------------------------------------
statistic.........      rate.per.second.............. total....... average......
name..............      max..... cur..... avg....... count....... per.trans....
AIJ file writes               0        0        0.0            0           0.0
    data                      0        0        0.0            0           0.0
    control                   0        0        0.0            0           0.0
    file extend               0        0        0.0            0           0.0
    switch over               0        0        0.0            0           0.0
records written               0        0        0.0            1           0.3
blocks written                0        0        0.0            0           0.0
    filler bytes              0        0        0.0            0           0.0
lock rebuilds                 0        0        0.0            0           0.0
    AIJ file reads            0        0        0.0            1           0.3
-------------------------------------------------------------------------------
Exit Graph Help Menu Options Pause Reset Set_rate Time_plot Unreset Write X_plot
```

画面の各フィールドについては、Performance Monitor のヘルプを参照してください。

4.1.1.9 Performance Monitor の「AIJ Journal Information」画面

「AIJ Journal Information」画面は、現行ノードにあるデータベースの After-image ジャーナルすべてに関するオンライン情報を表示します。画面に表示される内容は、ほとんどがリアルタイムで収集されるので、AIJ データベース・パラメータの変更や、バックアップやジャーナルの切替えなどの AIJ 操作を実行するたびに、画面は自動的に更新されます。

「AIJ Journal Information」画面はリアルタイム情報を表示するので、OUTPUT 修飾子を指定してバイナリ出力ファイルへ出力を記録することはできません。したがって、INPUT 修飾子を指定してバイナリ・ファイルを再生しても、この画面は参照できません。「AIJ Journal Information」画面は、「Journaling Information」サブメニューから選択できます。次に、「AIJ Journal Information」画面の例を示します。

```
Node: TRIXIE          Oracle Rdb V7.0-00 Performance Monitor 24-MAY-1996 13:40:22
Rate: 3.00 Seconds           AIJ Journal Information          Elapsed: 03:38:13.55
Page: 1 of 3          SQL_DISK1:[RICK.V70]MF_PERSONNEL.RDB;1          Mode: Online
--------------------------------------------------------------------------------
Journaling: Enabled   Shutdown:   60  Notify: Disabled  State: Accessible
ALS: Manual     ABS: Disabled  ACE: Disabled  FC: Enabled   CTJ: Enabled
After-Image.Journal.Name....... SeqNum AIJsize CurrEOF Status. State.......
RICK1                            Unused     512   Empty Latent  Accessible
RICK2                            Unused     512   Empty Latent  Accessible
RICK3                            Unused     512   Empty Latent  Accessible
RICK4                               19     513       2 Current Accessible
RICK5                            Unused     512   Empty Latent  Accessible
RICK6                               18 *BACKUP NEEDED* Written Accessible
Available AIJ slot 1
Available AIJ slot 2
Available AIJ slot 3
Available AIJ slot 4
Available AIJ slot 5
Available AIJ slot 6
Available AIJ slot 7
Available AIJ slot 8
--------------------------------------------------------------------------------
Bell Exit Help Menu >next_page <prev_page Refresh Set_rate Write Zoom !
```

「AIJ Journal Information」画面には、AIJ に関する一般的な情報や、予約済み AIJ スロットなど、個々のジャーナルに関する情報を表示します。

「AIJ Journal Information」画面で表示されるのは、現行ノードのみに関する内容なので注意してください。データベースを変更するあらゆるノードが After-image ジャーナルにアクセスするので、画面上の情報は古くなっている可能性があります。したがって、「AIJ Journal Information」画面下の水平メニューには「Refresh」オプションが提供されています。「Refresh」オプションを選択すると、現行ノードの情報は、データベースにアクセスしているその他すべてのノードと同期されます。

「AIJ Journal Information」画面の各フィールドについては、Performance Monitor のヘルプを参照してください。

After-image ジャーナルの詳細は、『Oracle Rdb7 Guide to Database Maintenance 』を参照してください。

4.1.1.10 Performance Monitor の「Snapshot Statistics」画面

「Snapshot Statistics」画面は、更新および読取り専用トランザクションのスナップショット・アクティビティを示します。次に、「Snapshot Statistics」画面の例を示します。

```
Node: TRIXIE          Oracle Rdb V7.0-00 Performance Monitor 24-MAY-1996 13:46:51
Rate: 3.00 Seconds             Snapshot Statistics           Elapsed: 03:44:42.12
Page: 1 of 1    RDBVMS_USER1:[LOGAN.MF_SAMPLE]MF_PERSONNEL.RDB;1    Mode: Online
------------------------------------------------------------------------------
statistic.........      rate.per.second.............. total....... average......
name..............      max..... cur..... avg....... count....... per.trans....
Total transactions         1        0        0.1            6           1.0
R/O transactions           0        0        0.1            4           0.7
retrieved record           7        0        2.9          210          35.0
  fetched line             7        0        2.9          210          35.0
    read snap page         0        0        0.0            0           0.0
stored snap record         0        0        0.0            0           0.0
    page in use            0        0        0.0            0           0.0
    page too full          0        0        0.0            0           0.0
    page conflict          0        0        0.0            0           0.0
    extended file          0        0        0.0            0           0.0
------------------------------------------------------------------------------
Exit Graph Help Menu Options Pause Reset Set_rate Time_plot Unreset Write X_plot
```

「Snapshot Statistics」画面の各フィールドについては、Performance Monitor のヘルプを参照してください。

4.1.1.11 Performance Monitor の「Checkpoint Statistics」画面

「Checkpoint Statistics」画面は、トランザクションとチェックポイント・アクティビティを示します。特定のデータベースのノードで実行されているすべてのプロセスに関する統計が表示されます。画面では、チェックポイントの総数が、チェックポイントが発生した原因ごとに分類して表示されます。また、チェックポイント間隔の合計が、3 つの測定方法を使って表示されます。この数値から、チェックポイントの平均間隔を計算できるので、チェックポイント間隔をどの程度調整すればよいかわかります。

「Checkpoint Statistics」画面には、秒ごとやトランザクションごとのイベントの測定値を示す列があります。この列に表示される数値は、I/O 操作など頻繁に発生するイベントの測定に便利です。一般的に、チェックポイントが発生する頻度はこれよりも低いため、「total count」の数値の方が参考になります。チェックポイント統計を使って最適なチェックポイント間隔を決定するには、この列に表示される値を参考にすることをお薦めします。

「Checkpoint Statistics」画面は、「Journaling Information」サブメニューから選択できます。次に、「Checkpoint Statistics」画面の例を示します。

```
Node: TRIXIE          Oracle Rdb V7.0-00 Performance Monitor 24-MAY-1996 14:43:06
Rate: 3.00 Seconds             Checkpoint Statistics          Elapsed: 04:40:57.33
Page: 1 of 1          SQL_DISK1:[RICK.V70]MF_PERSONNEL.RDB;1          Mode: Online
------------------------------------------------------------------------------
statistic.........      rate.per.second.............. total....... average......
name..............      max..... cur..... avg....... count....... per.trans....
transactions                  6        4        2.8          373           1.0
checkpoints                   1        0        0.1           16           0.0
    AIJ growth                1        0        0.1           16           0.0
    txn limit                 0        0        0.0            0           0.0
    time limit                0        0        0.0            0           0.0
    rollback                  0        0        0.0            0           0.0
    AIJ backup                0        0        0.0            0           0.0
    global                    0        0        0.0            0           0.0
interval: AIJ blks           28        0        3.2          423           1.1
interval: tx count           24        0        2.7          362           1.0
interval: seconds            11        0        0.6           87           0.2
checkpoint stall              0        0        0.0            0           0.0
flushed buffers               0        0        0.0            0           0.0
------------------------------------------------------------------------------
Exit Graph Help Menu Options Pause Reset Set_rate Time_plot Unreset Write X_plot
```

「Checkpoint Statistics」画面の各フィールドについては、Performance Monitor のヘルプを参照してください。

チェックポイント処理は、リカバリ時間に影響を与えるので注意してください。チェックポイント統計を使用する最大の目的は、チェックポイントごとの平均間隔を計算することにあります。この平均値は、total count 列の数値を使って計算できます。チェックポイントの各種類の原因について、チェックポイントごとの平均間隔に基づき、チェックポイント間隔をどの程度調整すればよいか判断してください。

チェックポイントには、次のような上限を設定できます。

- .aij ファイルのサイズ伸長
  .aij ファイルのサイズ伸長の上限を設定するには、ALTER DATABASE 文の CHECKPOINT INTERVAL IS n BLOCKS 句を使用します。CHECKPOINT INTERVAL IS 1000 BLOCKS を指定すると、プロセスの最後のチェックポイントの後、.aij ファイルのサイズが 1000 ブロック以上大きくなったときに、各プロセスはチェックポイント処理を実行します。
- 時間
  時間の上限を指定するには、ALTER DATABASE 文で CHECKPOINT TIMED EVERY n SECONDS 句を使用します。CHECKPOINT TIMED EVERY 600 SECONDS を指定する

と、最後のチェックポイントの後、10 分以上経過したときに、各プロセスはチェックポイント処理を実行します。

- トランザクション
  トランザクションの上限を指定するには、RDM$BIND_CKPT_TRANS_INTERVAL 論理名または RDB_BIND_CKPT_TRANS_INTERVAL 構成パラメータを使用します。RDM$BIND_CKPT_TRANS_INTERVAL をシステム論理セットとして 10 に設定すると、ユーザーが RDM$BIND_CKPT_TRANS_INTERVAL 論理名を他の値に再定義しない限り、トランザクションが 10 件を超えた時点で、各プロセスはチェックポイント処理を実行します。つまり、RDM$BIND_CKPT_TRANS_INTERVAL をプロセス論理名としてユーザー定義し、5 に設定すると、トランザクションが 5 件を超えたときに、ユーザーはチェックポイントを実行します。

上の例では、すべてのチェックポイントは .aij ファイル・サイズ伸長の上限によってトリガーされています。これは、トランザクションの上限が高すぎる、または .aij ファイル・サイズ伸長の上限が低すぎることを意味します。この項では、チェックポイント・リミットの種類ごとに、チェックポイントの平均間隔を計算する方法を説明します。各チェックポイント・リミットについて、チェックポイントごとの平均間隔を計算したら、これに近い値でチェックポイントがトリガーされるように、上限値を設定してください。これにより、パフォーマンスを最適化できます。

チェックポイントの平均間隔を計算するには、間隔の各カテゴリをチェックポイントの合計で割ってください。上の例では、秒間隔のカウントが 87、チェックポイントの合計が 16 なので、時間によってトリガーされるチェックポイント間の平均秒数は、およそ 5 です。

同様に、トランザクション間隔は 362、チェックポイントの合計が 16 なので、チェックポイント間の平均トランザクション数は、およそ 23 です。

ただし、AIJ ブロック単位で平均間隔を計算する場合、AIJ ブロック間隔をチェックポイントの合計から、AIJ バックアップによるカウントを差し引いた値で割ってください。AIJ バックアップで発生したチェックポイントは、チェックポイントの合計にカウントされていますが、AIJ ブロック間隔の合計にはカウントされていません。次の式から、.aij ファイルのサイズは、チェックポイントごとに、平均でおよそ 26 ブロック増大することがわかります。
平均 AIJ ブロック間隔 = 423 / (16 - 0) = 26.4375

チェックポイントの合計カウントと原因別のカウントは、1 対 1 に対応しない点に注意してください。これは、複数のイベントが 1 つのチェックポイントをトリガーする可能性があるからです。たとえば、時間と .aij ファイル・サイズ伸長の両方が原因となって、チェックポイントが発生することがあります。この列の数値は両方とも 1 ずつ増加しますが、チェックポイント合計の列の数値は 1 しか増加しません。

4.1.1.12 Performance Monitor の「Checkpoint Information」画面

「Checkpoint Information」画面には、プロセス・チェックポイント情報が示されます。現行ノードにあるデータベースに接続しているプロセスについて、各プロセスを 1 行で表示しま

す。プロセスがデータベースから切断されると、空白行が表示されます。プロセスがデータベースに再度接続すると、切断するまでは、画面の同じ位置に表示されます。

「Checkpoint Information」画面は、「Process Information」サブメニューから選択できます。次は、「Checkpoint Information」画面の例です。

```
Node: TRIXIE          Oracle Rdb V7.0-00 Performance Monitor  9-NOV-1995 10:08:36
Rate: 1.00 Second       Checkpoint Information (Unsorted)    Elapsed: 00:16:06.99
Page: 1 of 1       KODD_TEST:[R_ANDERSON.OE_MASTER]OE_RDB.RDB;1      Mode: Online
--------------------------------------------------------------------------------
Process.ID  Ckpt.Vno:Ckpt.Vbn  QuietVno  Tx.Start.Time  AIJ: 2:8148
2083CE80:1s         2     7261
20819292:1s
20814D51:1          2     6251          2  10:07:29.76
20821752:1          2     6322          2  10:07:28.57
2082D156:1          2     7618          2  10:08:06.90
20822D57:1          2     6677          2  10:07:12.16
2081875B:1          2     7258          2  10:07:37.89
--------------------------------------------------------------------------------
Config Exit Help Menu >next_page <prev_page Refresh Set_rate Write Zoom !
```

「Checkpoint Information」画面の各フィールドと各種構成オプションについては、Performance Monitor のヘルプを参照してください。

4.1.2 バッファの管理

バッファとは、読取りと更新の操作の間、データベース・ページを一時的に格納するために使用する内部メモリー領域です。データベースからデータを要求すると、Oracle Rdb は、バッファに格納できるだけのデータベース・ページをメモリーへ読み込みます。

データベースの作成または変更では、次の 2 通りの方法で、データベース・ページ用のバッファを設定できます。

- ユーザー専用のバッファ（ローカル・バッファ）を選択。これは、デフォルトです。
- 各ノードで共通のバッファ・プール（グローバル・バッファ）を選択。

ローカル・バッファを選択すると、Oracle Rdb は、各ユーザー・プロセスにローカル・バッファ・プールを提供します。複数のプロセスが同じページを使用する場合、プロセスが同じノード上で実行されていても、各プロセスは、ディスクから専用のローカル・バッファ・プールへページを読み込みます。

グローバル・バッファを選択すると、Oracle Rdb は、各データベースに 1 つのグローバル・バッファ・プールを提供します。マルチノード・システムでは、各ノードがデータベースごとに専用のグローバル・バッファ・プールを持ち、異なるノード上のプールは協調して動作します。ディスクからグローバル・バッファ・プールへページを読み込めるのは、一度に 1 つのプロセスのみですが、グローバル・バッファ・プール内のページへのアクセスは、同時に複数のプロセスが実行できます。

一般的に、データベース内でデータを共有する範囲を考えて、ローカル・バッファまたはグローバル・バッファを選択してください。

- 多くのプロセスが同じページへ頻繁にアクセスする場合は、グローバル・バッファを有効にすると、I/O を減らし、メモリーを効率的に使用できるので、パフォーマンスは向上します。
- プロセスが同じページに何度もアクセスする場合は、グローバル・バッファを有効にすると、パフォーマンスを向上できます。
- 各プロセスが専用のページにアクセスする場合には、グローバル・バッファを有効にしても、パフォーマンスは向上しません。

この項とこのマニュアルで使用する、ユーザーやプロセスという用語は、データベースへの 1 つの接続を指します。Oracle Rdb は、1 つのプロセスからの複数の接続を、複数ユーザーとみなします。したがって、1 つのユーザー・プロセスまたはアプリケーションがデータベースに 10 回接続する場合は、1 つのユーザー・プロセスまたはアプリケーションが 1 回の接続で使用するデータベース・リソースの 10 倍を消費することになります。例 4-1（4-20）は、プロセス ID2080F11A のユーザー・プロセス SMITH がデータベースへ 1 回アクセスした状態で、RMU Show Users コマンドを実行すると、プロセス 2080F11A に 20 のグローバル・バッファが割り当てられていることを示しています。プロセス ID 20808D50 のユーザー・プロセス RICK が、データベースに 2 回接続した状態で、RMU Show Users コマンドを実行すると、それぞれの接続に対して 20 のバッファが割り当てられ、これが 1 つのプロセスに割り当てられていることがわかります。データベース・リソースの消費が予測よりも早い場合は、RMU Show Users コマンドを実行し、1 つ以上のプロセスがデータベースに複数回数接続していないかどうか確認してください。

例 4-1 プロセスがデータベースに接続するごとに、1 プロセスに割り当てられたリソースを受け取る

```
SQL> -- User SMITH attaches to the database:
SQL> ATTACH 'FILENAME mf_personnel';
SQL> --
SQL> -- Show that the user process SMITH is allocated 20 global
SQL> -- buffers when attached to the database:
Oracle Rdb V7.0-00 on node TRIXIE 28-MAY-1996 09:22:18.51
database _$111$DUA176:[LOGAN.V70]MF_PERSONNEL.RDB;1
    - First opened 28-MAY-1996 09:21:53.86
    - current after-image journal file is
_$111$DUA176:[LOGAN.V70]RICK1.AIJ
;1
    - global buffer count is 1000
    - maximum global buffer count per user is 40
    - 980 global buffers free
    - 1 active database user
    - 2080F11A:1 - SMITH - non-utility, SMITH - active user
        - image $111$DUA600:[SQL_X07001.VAX.][CODE]SQL$701.EXE;1
```

```
        - 20 global buffers allocated
SQL> --
SQL> -- User process RICK from the same node attaches to the database
SQL> -- twice and each attach is allocated 20 global buffers:
SQL> ATTACH 'ALIAS MF_PERS1 FILENAME mf_personnel';
SQL> CONNECT TO 'ALIAS MF_PERS1 FILENAME mf_personnel' AS 'TEST';
SQL> $ RMU/SHOW USERS mf_personnel
Oracle Rdb V7.0-00 on node TRIXIE 28-MAY-1996 09:27:38.59
database _$111$DUA176:[LOGAN.V70]MF_PERSONNEL.RDB;1
    - First opened 28-MAY-1996 09:21:53.86
    - current after-image journal file is
_$111$DUA176:[LOGAN.V70]RICK1.AIJ
;1
    - global buffer count is 1000
    - maximum global buffer count per user is 40
    - 940 global buffers free
    - 3 active database users
    - 2080F11A:1 - SMITH - non-utility, SMITH - active user
        - image $111$DUA600:[SQL_X07001.VAX.][CODE]SQL$701.EXE;1
        - 20 global buffers allocated
    - 20808D50:1 - _RTA7: - non-utility, RICK - active user
        - image $111$DUA600:[SQL_X07001.VAX.][CODE]SQL$701.EXE;1
        - 20 global buffers allocated
    - 20808D50:2 - _RTA4: - non-utility, RICK - active user
        - image $111$DUA600:[SQL_X07001.VAX.][CODE]SQL$701.EXE;1
        - 20 global buffers allocated
SQL>
```

例 4-2（4-21）の Performance Monitor の「Buffer Information」画面と SQL SHOW DATABASE の出力は、Oracle Rdb のローカル・バッファとグローバル・バッファのパラメータを示します。

例 4-2 ローカルおよびグローバル・バッファ・パラメータ

```
Node: TRIXIE         Oracle Rdb V7.0-00 Performance Monitor 21-JUN-1996 11:59:07
Rate: 3.00 Seconds             Buffer Information           Elapsed: 00:00:07.47
Page: 1 of 1       RDBVMS_USER1:[LOGAN.V70]MF_PERSONNEL.RDB;1       Mode: Online
--------------------------------------------------------------------------------
Default user buffer count is 20 (1)
Default recovery buffer count is 20
Buffer size is 6 blocks (2)
Global Buffers are disabled (3)
- Global buffer count is 250 (4)
- Maximum global buffer count per user is 5 (5)
- Page transfer via memory is disabled
Global section size with global buffers disabled is 87600 bytes
- With global buffers enabled is 997430 bytes
Asynchronous pre-fetch is enabled
```

```
- Maximum pre-fetch depth is 5 buffers
Detected asynchronous pre-fetch is enabled
- Maximum pre-fetch depth is 4 buffers
- Pre-fetch threshold is 4 pages
Asynchronous batch-write is enabled
- Clean buffer count is 5
- Maximum batch size is 4 buffers
--------------------------------------------------------------------------------
Exit Help Menu Options Refresh Set_rate Write !
.
.
.
$ SQL
SQL> ATTACH 'FILENAME mf_personnel';
SQL> SHOW DATABASE *
Default alias:
    Oracle Rdb database in file MF_PERSONNEL
        Multischema mode is disabled
        Number of users:               50
        Number of nodes:               16
        Buffer Size (blocks/buffer):   6 (2)
        Number of Buffers:             20 (1)
        Number of Recovery Buffers:    20
        Snapshots are Enabled Immediate
        Carry over locks are enabled
        Lock timeout interval is 0 seconds
        Adjustable lock granularity is enabled
        Global buffers are disabled (3) (number is 250, (4) user limit is 5 (5)
.
.
.
SQL>
```

次に、例 4-2（4-21）で示した番号に対応するバッファ・パラメータについて説明します。

(1) NUMBER OF BUFFERS パラメータ
NUMBER OF BUFFERS パラメータのルート・ファイルの値は、データベースに接続する各プロセスに割り当てられたバッファの数のデフォルト値です。他のバッファ・パラメータの設定や、ローカル・バッファとグローバル・バッファのどちらを有効にするかによって、プロセスがデータベースに接続したとき、このパラメータで指定した数のバッファがプロセスに割り当てられない場合もあります。NUMBER OF BUFFERS パラメータの詳細は、4.1.2.2 項（4-25）を参照してください。

(2) BUFFER SIZE パラメータ
バッファあたりのブロック数を指定します。BUFFER SIZE パラメータで指定した値は、ローカル・バッファとグローバル・バッファのいずれにも有効です。この値は、データ

ベース・ルート・ファイルに格納されます。BUFFER SIZE パラメータの詳細は、4.1.2.1 項（4-23）を参照してください。

(3) グローバル・バッファは、無効化または有効化するパラメータです。
ローカル・バッファは、デフォルトで有効にされています（グローバル・バッファは、デフォルトで無効にされています）。データベースでローカル・バッファを有効にすると、NUMBER IS パラメータ（4）と USER LIMIT パラメータ（5）で指定した値は、適用されません。このパラメータの値が適用されるのは、グローバル・バッファを有効にしたときのみです。グローバル・バッファの有効化の詳細は、4.1.2.6 項（4-34）を参照してください。

(4) NUMBER IS パラメータ
NUMBER IS パラメータで指定したアクティブな値は、データベースのノードあたりのグローバル・バッファの合計です。NUMBER IS パラメータの詳細は、4.1.2.7 項（4-35）を参照してください。

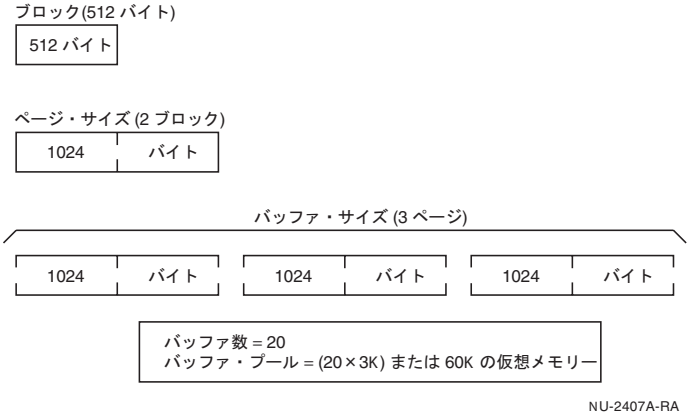
(5) USER LIMIT パラメータ
USER LIMIT パラメータで指定したアクティブな値は、データベースに接続しているプロセスに割当て可能なグローバル・バッファの最大数です。
例 4-2（4-21）では、mf_personnel データベースの USER LIMIT パラメータの値は、NUMBER OF BUFFERS パラメータの値よりも小さくなっています。通常は、グローバル・バッファが有効化されている場合、USER LIMIT パラメータには、NUMBER OF BUFFERS よりも大きな値に設定してください。USER LIMIT パラメータの詳細は、4.1.2.8 項（4-38）を参照してください。

例 4-2（4-21）で示すパラメータの値は、データベース・ルート・ファイルに格納されています。NUMBER IS および USER LIMIT パラメータは、RMU Open コマンドを実行すれば、ルート・ファイルと異なる値を指定できます。RMU Open コマンドを使って NUMBER IS および USER LIMIT パラメータを設定する詳細は、4.1.2.7 項（4-35）および 4.1.2.8 項（4-38）を参照してください。

4.1.2.1 バッファ・サイズの指定

SQL CREATE DATABASE または ALTER DATABASE 文の BUFFER SIZE パラメータで指定した値は、1 つのバッファに格納できるデータベース・ページの数を示します。この値は、ローカル・バッファとグローバル・バッファのいずれにも有効です。デフォルト値は、ページ・サイズの 3 倍です。デフォルトのページ・サイズを使用する場合、デフォルトのバッファ・サイズは 6 ブロック（デフォルトのページ・サイズである 2 ブロックの 3 倍）、つまり 3072 バイトとなります。図 4-1（4-24）は、BLOCK SIZE、PAGE SIZE、BUFFER SIZE のデフォルト値の関係を示しています。

図 4-1 バッファ・プール：データベース・パラメータのデフォルト値



選択したバッファ・サイズは、パフォーマンスに影響を与える可能性があり、どのようなデータベース操作が多く発生するかによって異なります。バッファ・サイズに大きな値を指定すると、バッファに格納されている近接ページを読み出すので、ある程度のリードアヘッド機能を発揮でき、順次読取りでの I/O 操作を軽減できます。

ただし、ランダムなデータ読出しでは、バッファ・サイズに小さな値を指定した方が効果的です。必要なページがバッファ内に存在しない場合、Oracle Rdb は、ディスクからバッファ・サイズだけのページを読み込む必要があります。バッファ・サイズが大きいと、1 つのバッファをいっぱいにするために、多くのブロックを読み込まなければなりません。

バッファ・サイズは、データベース全体にわたるパラメータです。ページおよびバッファあたりのブロック数には、64 ブロック未満という制限があります。ただし、マルチファイル・データベースでは、記憶領域によってページ・サイズが異なることがあるので、ページ・サイズは、各記憶領域に格納されるレコードのサイズによって決まります。データベースを作成した後にバッファ・サイズを変更するには、SQL EXPORT および IMPORT 文を使用するか、ALTER DATABASE 文の BUFFER SIZE IS 句を使用します。SQL 文の詳細は、『Oracle Rdb7 SQL Reference Manual』を参照してください。

BUFFER SIZE パラメータには、マルチファイル・データベースのすべての記憶領域について、どのページ・サイズでも割り切れる値を設定してください。これにより、メモリーの無駄使いがなくなります。たとえば、3 つの記憶領域があり、ページ・サイズはそれぞれ 16、24、32 ブロックだとすると、バッファの使用を最適化するには、バッファ・サイズに 96 ブロックを選択してください。バッファ・サイズを 64 ブロックにすると、ページ・サイズが 24 ブロックの記憶領域では、バッファ 1 つあたり 16 ブロック（25%）が無駄になってしまいます。Oracle Rdb は、バッファに格納できるだけのページを読み取るので、上の例では 16 ブロックが残り、無駄になります。

ページ・サイズの定義や変更で、バッファ・サイズを超える値を指定すると、Oracle Rdb は次のエラー・メッセージを返します。

```
RDMS-F-BUFSMLPAG, buffer size is less than page size.
```

4.1.2.2 ユーザー・バッファのデフォルト数の指定

NUMBER OF BUFFERS パラメータで指定した値は、データベースに接続しているユーザー・プロセスに Oracle Rdb が割り当てるバッファの数のデフォルト値です。他のバッファ・パラメータ（RDM$BIND_BUFFERS 論理名や RDB_BIND_BUFFERS 構成パラメータで指定するバッファの数など）や、ローカル・バッファとグローバル・バッファのどちらを有効にするかによっては、データベースへの接続時に、NUMBER OF BUFFERS パラメータで指定した数のバッファがプロセスに割り当てられない場合があります。

注意：NUMBER OF BUFFERS パラメータとグローバル・バッファ・パラメータ NUMBER IS の違いに注意してください。NUMBER IS パラメータは、グローバル・バッファ・プール内にあるバッファの合計数を指定するものであり、適用されるのは、グローバル・バッファを有効にしたときのみです。グローバル・バッファの有効化については、4.1.2.6 項（4-34）を参照してください。

NUMBER OF BUFFERS パラメータの指定には、SQL ALTER DATABASE 文と CREATE DATABASE 文の NUMBER OF BUFFERS 句を使用します。

```
SQL> ALTER DATABASE FILENAME mf_personnel
  1> NUMBER OF BUFFERS IS m;
```

また、NUMBER OF BUFFERS パラメータの指定には、RMU Restore コマンドの Local_Buffers=(Number=m) 修飾子も使用できます。

```
$ RMU/RESTORE/LOCAL_BUFFERS=(NUMBER=m)/NOCDD mf_pers_backup
$ rmu -restore -local_buffers=¥(number=m¥) mf_pers_backup
```

ALTER DATABASE 文と CREATE DATABASE 文または RMU Restore コマンドを使って NUMBER OF BUFFERS パラメータを指定すると、有効化されているのがローカル・バッファかグローバル・バッファかに関係なく、指定した値はデータベース・ルート・ファイルに格納されます。

また、論理名 RDM$BIND_BUFFERS または構成パラメータ RDB_BIND_BUFFERS では、プロセスがデータベースに接続したときに割当て可能なバッファの数を指定できます（NUMBER OF BUFFERS パラメータの指定とは異なる値を指定したい場合）。
RDM$BIND_BUFFERS 論理名または RDB_BIND_BUFFERS 構成パラメータには、有効な値として、正の整数を指定してください。たとえば、次のコマンドを発行するプロセスは、データベースへの接続時に 35 のバッファの割当てを要求します。

```
$ DEFINE RDM$BIND_BUFFERS 35
```

次に、Oracle Rdb では、データベースへの接続時にユーザー・プロセスに割り当てるバッファの数を、ローカル・バッファおよびグローバル・バッファ・パラメータの値に基づいてどのように決定するかを説明します。

- ローカル・バッファが有効化されている場合、プロセスが、RDM$BIND_BUFFERS 論理名や RDB_BIND_BUFFERS 構成パラメータで異なる値を指定している場合を除いて、Oracle Rdb は、データベースに接続している各プロセスに対して、NUMBER OF BUFFERS パラメータで指定した数のバッファ（ルート・ファイルに格納）を割り当てます。RDM$BIND_BUFFERS 論理名または RDB_BIND_BUFFERS 構成パラメータで異なる値が指定されている場合は、その数のバッファを割り当てます。
- グローバル・バッファが有効化されている場合、Oracle Rdb は、次のパラメータの値に基づいて、プロセスに割り当てるバッファの数を決定します。
  - USER LIMIT パラメータ
    アクティブな USER LIMIT パラメータの値は、データベースを使用しているプロセスに Oracle Rdb が割当て可能なグローバル・バッファの最大数です。USER LIMIT パラメータには、次の 2 通りの設定方法があります。最初の方法では、SQL CREATE DATABASE 文または ALTER DATABASE 文、または RMU Restore Global_Buffers=(User_Limit=t) コマンドを使用します。指定した値は、データベース・ルート・ファイルに格納されます。もう 1 つは、RMU Open Global_Buffers=(User_Limit=t) コマンドを使用する方法です。このコマンドは、データベースをオープンし、オープンしている間に割当て可能なバッファの最大数を指定します。USER LIMIT パラメータを指定して RMU Open Global_Buffers コマンドを実行すると、データベースをクローズするまでの間、データベース・ルート・ファイルに格納されている USER LIMIT の値は上書きされます。RMU Show Users database-name コマンドは、USER LIMIT パラメータのアクティブな値（現行ノード上のデータベースで現在有効な値）を表示します。
    USER LIMIT パラメータの詳細と設定方法の例は、4.1.2.8 項（4-38）を参照してください。
  - RDM$BIND_BUFFERS 論理名または RDB_BIND_BUFFERS 構成パラメータ
    RDM$BIND_BUFFERS 論理名または RDB_BIND_BUFFERS 構成パラメータがプロセスで定義されている場合、その値が、アクティブな USER LIMIT の値より小さければ、Oracle Rdb は、RDM$BIND_BUFFERS 論理名または RDB_BIND_BUFFERS 構成パラメータで指定されている数のバッファを割り当てます。
    RDM$BIND_BUFFERS または RDB_BIND_BUFFERS の値がアクティブな USER LIMIT の値よりも大きい場合、プロセスには、アクティブな USER LIMIT で指定されている数のバッファを割り当てます。
  - NUMBER OF BUFFERS パラメータ
    RDM$BIND_BUFFERS 論理名または RDB_BIND_BUFFERS 構成パラメータがプロセスで定義されていない場合、NUMBER OF BUFFERS の値がアクティブな USER

LIMIT の値より小さければ、Oracle Rdb は、NUMBER OF BUFFERS パラメータで指定した数のバッファをプロセスに割り当てます。NUMBER OF BUFFERS パラメータの値がアクティブな USER LIMIT の値よりも大きい場合、アクティブな USER LIMIT で指定した数のバッファをプロセスに割り当てます。

RDM$BIND_BUFFERS 論理名または RDB_BIND_BUFFERS 構成パラメータでは、個々のプロセスのニーズに合わせた設定ができます。たとえば、夜間に実行する大きなバッチ・ジョブには、日中実行する対話型プロセスよりも多くのバッファが必要になります。バッチ・ジョブ・プロセス向けに RDM$BIND_BUFFERS または RDB_BIND_BUFFERS を定義すれば、バッチ・ジョブに割り当てるバッファの数だけを増やし（グローバル・バッファが有効化され、USER LIMIT パラメータのアクティブな値が論理名 RDM$BIND_BUFFERS または構成パラメータ RDB_BIND_BUFFERS で指定した値よりも小さい場合を除く）、データベースにアクセスする他のプロセスに割り当てるバッファは、デフォルト値のままにすることができます。デフォルトでは、RDM$BIND_BUFFERS 論理名または RDB_BIND_BUFFERS 構成パラメータで指定した値は、定義しているプロセス固有であり、このプロセスが使用するすべてのデータベースに適用される点に注意してください。

OpenVMS VAX OpenVMS Alpha

必要な権限がある場合には、グループまたはシステム論理名として、RDM$BIND_BUFFERS を定義することにより、さらに大きなプロセス群の設定ができます。◆

バッファの数を増やす利点の 1 つとして、バッファ・プールに格納できるインデックス・ノードの数が増えることがあります。結果として、I/O 操作の数が減少します。また、コミットまで遅延する出力が増え、非同期 I/O 操作を実行するため、応答時間も向上します。特に、マルチユーザー・データベースでは効果的です。

NUMBER OF BUFFERS パラメータを指定しない場合、デフォルトのバッファ数は 20 です。現在のバッファ・プールに存在しないデータをユーザー・プロセスが要求すると、Oracle Rdb はバッファを再利用し、更新済バッファをデータベースに書き込んで、新しいページを読み込めるようにスペースを作ります。

4.1.2.3 ローカル・バッファのチューニング

ローカル・バッファが有効化されている場合に、各データベース・ユーザーに割り当てるためバッファ・プールに予約されている仮想メモリーのサイズは、データベース・パラメータ BUFFER SIZE および NUMBER OF BUFFERS の値から計算できます。パラメータのデフォルト値（図 4-1（4-24）を参照）から、次のように計算できます。

バッファ・プール = NUMBER OF BUFFERS × BUFFER SIZE

バッファ・プール = 20 × (6 × 512) = 61440 バイト

したがって、各ユーザーには、61,440 バイト、120 ブロックのバッファ・プールが割り当てられます。バッファが実際に消費するメモリーに、バッファ管理によるオーバーヘッドが加わるため、バッファ・プールに必要な実際のサイズは、120 ブロックを若干上回ります。

まれですが、使用可能な物理メモリーやクォータに対して、バッファ・プールが大きすぎる場合、オペレーティング・システムは、データベース・ページの読取りに加えて、バッ

ファ・プールの仮想ページングを実行することもあります。これにより、パフォーマンスは低下します。ページングを避けるためには、1 つのノード上で同時にアクティブになるユーザーが必要とするメモリー・サイズの合計が、物理メモリーのサイズよりも小さくなるようにしてください。

バッファ・プールが小さすぎると、データベース・ページをメモリーに読み込むために必要な I/O 操作が多くなります。

データベースで必要なバッファ・サイズとバッファの数は、データベースの行サイズと、ユーザーが実行するデータベース操作のタイプから決定してください。

- BUFFER SIZE に大きな値を指定すると、1 回の I/O 操作で読み込めるデータの容量が増え、データが近くに格納されているので、隣接した行を検索できます。これは、順次データ読取りで効果的です。
- NUMBER OF BUFFERS パラメータを、デフォルト（20）よりも大きな値に指定すると、Oracle Rdb は、以前使用した行を、ユーザーのバッファ・プール内に保持する可能性が高くなります。これは、同じデータを複数回参照するアプリケーションに効果的です。インデックスは、同じデータを何度も参照するアプリケーションのよい例です。ソート済のインデックスで範囲検索を実行する場合、バッファの数に大きな値を指定するとよいでしょう。また、アプリケーションが同じデータを複数回参照するときは、グローバル・バッファを有効にしてもよいでしょう。

一般的に、トランザクション・タイプが混在している作業負荷では、デフォルト値でも十分なパフォーマンスを発揮します。ただし、一部のトランザクション・タイプの発生頻度が他よりも高い場合は、データベース・パラメータの値を調整することもできます。一般的には、小さなバッファが多数存在する場合、I/O 操作が減少し、バッファ・プールの仮想ページングを避けることができるので、大きなバッファが少数存在する場合よりも、パフォーマンスは高くなります。ローカル・バッファとメモリー消費の詳細は、8.1.3 項（8-15）を参照してください。

クエリーが、インデックス列をもとに行を取り出す場合、Oracle Rdb は、インデックスに格納されたデータベース・キーを使用して、行を直接検索します。ただし、Oracle Rdb は、行を含むページ全体に加えて、ユーザーのバッファ・プールにあるバッファがいっぱいになるだけのデータベース・ページを読み込みます。Oracle Rdb は、必ず、バッファがいっぱいになるだけのデータを、データベースからユーザーのバッファ・プールに読み込みます。

データベース内のデータの順次読取りでは、Oracle Rdb は、連続したページをデータベース・ファイルからユーザのバッファ・プールへ読み込みます。したがって、バッファ・サイズとバッファの数によって、1 回の I/O 操作で読み取り、使用可能になるデータのサイズが決まります。次にユーザーが要求した行がバッファ・プール内に存在しない場合、Oracle Rdb は、最近使用していないデータが格納されているユーザー・バッファをフラッシュします。ユーザーが行を変更した場合、その行を含むバッファは、データベース・ファイルに書き戻した後、使用可能になります。バッファのデータが変更されていなければ、バッファはすぐに使用できます。

Performance Monitor の「PIO Statistics--Data Fetches」画面と「PIO Statistics--SPAM Fetches」画面では、バッファの数がパフォーマンスに与える影響を表示でき、パフォーマンスを向上するためにはどのような調整が必要なのかを判断できます（例 8-8（8-19）と例 8-9（8-20）を参照）。

4.1.2.4 物理メモリーでのローカル・バッファのロック

OpenVMS Alpha

Oracle Rdb では、OpenVMS Alpha システムからのデータベース・アクセス向けに、パフォーマンスを向上する機能が提供されています。この機能は、OpenVMS バッファ・オブジェクトを使用して、Oracle Rdb ローカル・バッファを物理メモリー内にロックします。Oracle Rdb ローカル・バッファをメモリー内にロックすることにより、対称マルチプロセッシング（Symmetric MultiProcessing：SMP）の並列性とスケーリングを向上でき、OpenVMS のオーバーヘッドをなくすことで I/O パフォーマンスを向上します。

この機能を使うには、次のような構成が必要です。

- Oracle Rdb と OpenVMS Alpha バージョン 6.1 以上を Alpha ユニプロセッサまたは SMP システム上で実行。
- 論理名 RDM$BIND_BUFOBJ_ENABLED を任意の値に定義。
- OpenVMS のユーザー・クオータ BYTLM を、ローカル・バッファ・プールの数だけ増加。たとえば、ローカル・バッファの数が 1000、バッファ・サイズが 8 ブロックの場合、次の式から BYTLM パラメータの値を計算。
  1000 × (8 × 512) = 4,096,000 バイト

メモリーに制約がある場合、この機能は使用しないでください。バッファ・オブジェクトとして定義されている OpenVMS ページは、ページングやスワップを実行できますが、物理メモリー内に存在しなければなりません。イメージが終了するまでの間、他のプロセスはこの物理メモリーを使用できなくなります。

注意： Oracle Rdb では、グローバル・バッファを有効にしたデータベース上では、OpenVMS バッファ・オブジェクトを使用できません。

◆

4.1.2.5 グローバル・バッファ・プール

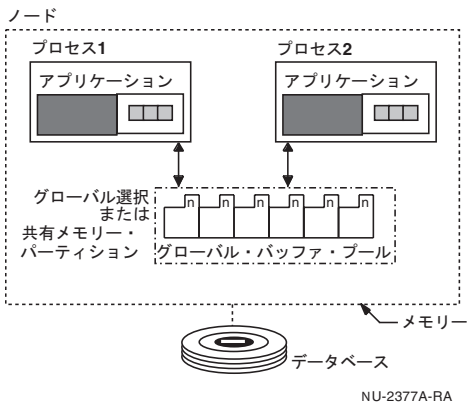
データベースでローカル・バッファを有効にすると、Oracle Rdb は、各ユーザー・プロセス用に別のバッファ・プールを管理します。データベースでグローバル・バッファを有効にすると、Oracle Rdb は、ユーザー・プロセスがデータベース・アクセスを実行する各ノード上のデータベース用に、グローバル・バッファ・プールを管理します。ノードからデータベースにアクセスしているすべてのユーザーは、そのノードのデータベース用に確保されたグローバル・バッファを使用します。したがって、データベースでグローバル・バッファを

有効にした場合、データベースのグローバル・バッファ・プールは、ユーザーがデータベース・アクセスを行う個々のノードのデータベース用グローバル・バッファ・プールで構成されます。

グローバル・バッファを有効にするには、SQL ALTER 文または CREATE DATABASE 文で GLOBAL BUFFERS ARE ENABLED 修飾子を指定するか、RMU Restore Global_Buffers=Enabled コマンドを実行します。グローバル・バッファの有効化の詳細は、4.1.2.6 項（4-34）を参照してください。

図 4-2（4-30）は、プロセスが 2 つある場合のグローバル・バッファ管理の例を示しています。ここでは、1 人ユーザーまたは 1 つのプロセスが 1 つの接続としてみなされます。Oracle Rdb は、1 つのプロセスからの複数の接続を、複数ユーザーとみなします。したがって、1 つのユーザー・プロセスがデータベースに 10 回接続する場合は、1 つのユーザー・プロセスまたはアプリケーションが 1 回の接続で使用するデータベース・リソースの 10 倍を消費することになります。接続は、それぞれが Oracle Rdb をコールしてデータベース・ページを要求します。次に、Oracle Rdb は、オペレーティング・システムをコールして、記憶デバイスからグローバル・バッファ・プールへページを読み込みます。

図 4-2 グローバル・バッファの管理



Oracle Rdb は、OpenVMS のグローバル・セクションと、Compaq Tru64 UNIX の共有メモリー・パーティションにグローバル・バッファ・プール（メモリー内にあるバッファのキャッシュ）を設定し、ページのバッファを読み込みます。ユーザー・プロセスは、グローバル・セクションまたは共有メモリー・パーティションを仮想メモリーへマッピングします。これにより、ユーザーが必要とする仮想メモリーのサイズは（ローカル・バッファを使用する場合よりも）大きくなりますが、物理メモリーを効率的に使用できます。グローバル・バッファ・プールには、各ページのコピーが 1 つだけありますが、ノード上で実行されている接続プロセスは、同じページを同時に読み取ることができます。データベースでグローバル・バッファを有効にすると、システムで必要となるグローバル・セクションまたは

共有メモリー・パーティションなどのリソースは増大します。また、Performance Monitor の「Buffer Information」画面では、データベースに対してグローバル・バッファを有効にする前に、グローバル・セクションまたは共有メモリー・パーティションのサイズを予測できます。詳細は、4.1.2.12 項（4-48）を参照してください。

プロセスはプールにある任意のバッファにアクセスできますが、特定の時点でプロセスが使用できるバッファの数には制限があり、これは割当てセットと呼ばれます。ユーザーの割当てセットにあるバッファの数は、ユーザーがデータベースに接続した時点で有効となっている次の値で決まります。

- NUMBER OF BUFFERS の値
- RDM$BIND_BUFFERS または RDB_BIND_BUFFERS の値
- USER LIMIT IS の値

4.1.2.2 項（4-25）では、ユーザー・プロセスの接続時に割り当てるバッファの数を、有効なローカルおよびグローバル・バッファ・パラメータをもとに、Oracle Rdb が判断する方法について説明します。

グローバル・バッファ・プールがいっぱいで、プロセスがプールにないページを要求した場合、最初の手順として、Oracle Rdb は、既存のグローバル・バッファを解放する必要があります。グローバル・バッファの解放では、Oracle Rdb は、最低使用頻度（Least-Recently Used: LRU）置換方針に改良を加えた方法を利用して、新しいページを空のグローバル・バッファに読み込みます。

ここでは、グローバル・バッファのしくみについて、簡単な例を挙げて説明します。

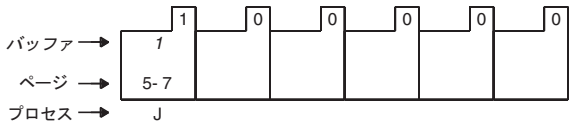
この例では、次のような内容を前提とします。

- 2 つのプロセスがデータベースに接続し、プロセッサは 3 つのバッファで構成される割当てセットを持ちます。
- すべてのページは、均一フォーマットの記憶領域から読み込まれます。記憶領域の最初のページは、SPAM ページです。SPAM ページとデータ・ページは、同時には読み取りません（Oracle Rdb は、SPAM ページとデータ・ページを同じバッファに読み込みません）。
- バッファ・サイズは 6 ブロック、ページ・サイズは 2 ブロックなので、各クランプは 3 ページです。クランプ内のどのページが要求される場合でも、1 クランプについて常に同じページ数を読み取ります。ユーザーがページ 2、3、4 のいずれを要求した場合でも、同じ 3 ページ（ページ 2 ～ 4）を読み取ります。
- グローバル・バッファを有効にし、グローバル・バッファ・プールには 6 つのバッファが存在します。

したがって、グローバル・プールは、合計 36 ディスク・ブロック（データベース・ページが 18）で構成されます。プロセスは、バッファ・プールにある 18 のページをすべて使用できますが、一度に参照できるのは 9 ページのみ（割当てセット内の 3 バッファ）です。

図 4-3（4-32）から図 4-8（4-34）は、J と T の 2 つのプロセスがグローバル・バッファにページを読み込む操作を示しています。プロセス J はクエリーを実行し、最初のグローバル・バッファに 3 ページが読み込まれます。複数のプロセスがページを参照できるので、Oracle Rdb は、各バッファにカウンタを設定し、バッファを参照しているプロセスの数を示します。図 4-3（4-32）では、カウンタは 1 にセットされ、バッファにアクセスしているプロセスがは 1 つであることを示しています。

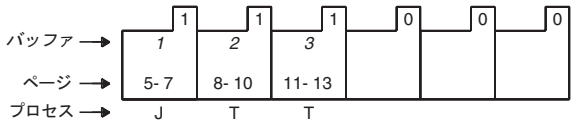
図 4-3 グローバル・バッファ・プール：バッファ 1



NU- 2378A- RA

プロセス T は処理を開始し、Oracle Rdb はさらに 6 ページをグローバル・バッファ・プールに読み込みます。これにより、図 4-4（4-32）で示すように、2 つのバッファがいっぱいになります。

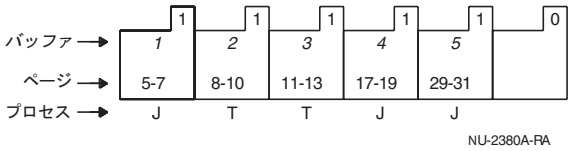
図 4-4 グローバル・バッファ・プールバッファ 1 ～ 3



NU- 2379A- RA

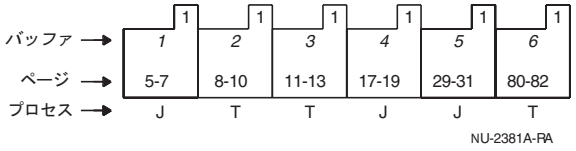
プロセス J は、Oracle Rdb に対して、さらに 6 ページをグローバル・バッファ・プールへ読み込むことを要求します。図 4-5（4-33）を参照してください。

図 4-5 グローバル・バッファ・プールバッファ 1 ～ 5



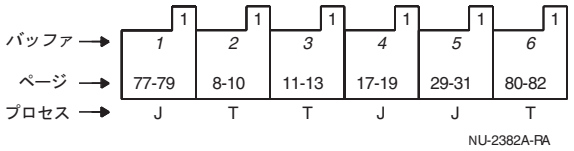
続けて、プロセス T がさらに 3 ページの読込みを要求したので、図 4-6（4-33）で示すように、グローバル・バッファ・プール内の残りのグローバル・バッファはいっぱいになります。

図 4-6 グローバル・バッファ・プールバッファ 1 ～ 6



プロセス J は、ページ 77 を読み取ろうとしますが、このページはバッファ・プールにはありません。プロセス J は、割当ての上限である 3 バッファに達しているため、ページ 77 をグローバル・バッファ・プールに読み込むためには、グローバル・バッファを解放する必要があります。プロセス J の割当てセット内のバッファに LRU ポリシーを適用すると、最初のグローバル・バッファをフラッシュし、追加の 3 ページが読み込まれます。この中の 1 ページは、要求されたページです。図 4-7（4-33）を参照してください。

図 4-7 グローバル・バッファ 1 の変更

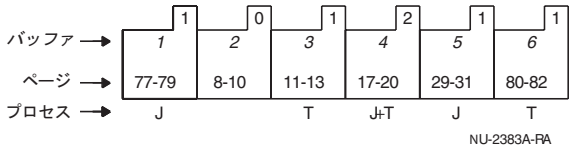


プロセス T は、ページ 17 を読み取ろうとしています。プロセス T は、割当ての上限である 3 バッファに達しているため、まずグローバル・バッファを解放する必要があります。プロセス T の割当てセットに LRU ポリシーを適用すると、プロセス T は、グローバル・バッファ 2 を解放し、カウンタは 0 にセットされます。ページ 8 ～ 10 を読み取るプロセスは存在しませんが、このページはグローバル・バッファ内に格納されます（図 4-8（4-34）を参

照)。プロセス T がグローバル・バッファ 2 のページを変更した場合、バッファを再割当てする前に、変更されたページをディスクに書き戻す必要があります。

プロセス J はページ 17 ～ 19 をグローバル・バッファ 4 に読み込んでいるため、プロセス T は、I/O を実行しなくても、ページ 17 を読み取ることができます。グローバル・バッファ 4 のページ範囲にアクセスしているユーザーは 2 人なので、グローバル・バッファ 4 のカウンタは 2 になります。図 4-8（4-34）を参照してください。

図 4-8 グローバル・バッファ 2 と 4 の変更



どのプロセスも、I/O を実行しないでページ 8 ～ 10 の読み取りを実行できます。ただし、現在グローバル・バッファ・プールに存在しないページをプロセスが要求すると、ページ 8 ～ 10 は、新しいページに置き換えられます。

4.1.2.6 グローバル・バッファの有効化

マルチノード・システム内のデータベースでグローバル・バッファを有効にすると、それぞれのノードが各データベースのグローバル・バッファ・プールを持ちます。データベースに対してグローバル・バッファを有効にするには、GLOBAL BUFFERS ARE ENABLED 句を指定して、SQL ALTER DATABASE 文と CREATE DATABASE 文を実行します。

```
SQL> ALTER DATABASE FILENAME mf_personnel
  1> GLOBAL BUFFERS ARE ENABLED;
```

グローバル・バッファの有効化は、データベースにユーザーが接続していない状態で実行してください。

グローバル・バッファを無効にするには、GLOBAL BUFFERS ARE DISABLED 句を指定して、SQL ALTER DATABASE 文と CREATE DATABASE 文を実行します。

```
SQL> ALTER DATABASE FILENAME mf_personnel
  1> GLOBAL BUFFERS ARE DISABLED;
```

グローバル・バッファの無効化は、データベースにユーザーが接続していない状態で実行してください。

また、グローバル・バッファの有効化は、RMU Restore コマンドで Global_Buffers=Enabled 修飾子を指定しても実行できます。

```
$ RMU/RESTORE/GLOBAL_BUFFERS=ENABLED/NOCDD mf_pers_backup
$ rmu -restore -global_buffers=enabled mf_pers_backup
```

グローバル・バッファの無効化は、RMU Restore コマンドで Global_Buffers=Disabled 修飾子を指定しても実行できます。

```
$ RMU/RESTORE/GLOBAL_BUFFERS=DISABLED/NOCDD mf_pers_backup
$ rmu -restore -global_buffers=disabled mf_pers_backup
```

ALTER DATABASE 文と CREATE DATABASE 文または RMU Restore コマンドを使用してグローバル・バッファの有効化または無効化を実行すると、設定内容はデータベース・ルート・ファイルに格納されます。

データベースでローカル・バッファを有効にすると、NUMBER IS と USER LIMIT パラメータで指定した値は適用されません。このグローバル・バッファ・パラメータの値が適用されるのは、グローバル・バッファを有効にしたときのみです。NUMBER OF BIFERS や BUFFER SIZE など他のバッファ・パラメータの値は、ローカル・バッファとグローバル・バッファのいずれにも有効です。

OpenVMS VAX OpenVMS Alpha

OpenVMS オペレーティング・システムでは、グローバル・セクションのサイズを変更できないので、接続ユーザーの増加に伴って、グローバル・バッファ・プールにバッファを追加することはできません。したがって、ユーザーあたりのバッファ・カウントのデフォルト値（NUMBER OF BUFFERS パラメータ）、グローバル・バッファの数（NUMBER IS パラメータ）、ユーザーあたりのバッファの最大数（USER LIMIT パラメータ）を適切に設定することが重要です。◆

グローバル・バッファは、グローバル・セクションや共有メモリー・パーティションなどのリソースを追加で必要とすることがあります。詳細は、4.1.2.12 項（4-48）を参照してください。

4.1.2.7 NUMBER IS パラメータ

NUMBER IS パラメータは、データベースのノードあたりのグローバル・バッファの数の合計です。NUMBER IS パラメータの指定には、SQL ALTER DATABASE 文と CREATE DATABASE 文の NUMBER IS 句を使用します。

```
SQL> ALTER DATABASE FILENAME mf_personnel
  1>    GLOBAL BUFFERS ARE ENABLED
  2>    (NUMBER IS n);
```

また、NUMBER IS パラメータの指定には、RMU Restore コマンドの Global_Buffers=Total=n 修飾子も使用できます。

```
$ RMU/RESTORE/GLOBAL_BUFFERS=(TOTAL=n)/NOCDD mf_pers_backup
$ rmu -restore -global_buffers=¥(total=n¥) mf_pers_backup
```

ALTER DATABASE 文と CREATE DATABASE 文または RMU Restore コマンドを使用して NUMBER IS パラメータを指定すると、設定値はデータベース・ルート・ファイルに格納さ

れます。この文やコマンドでは、クラスタ内で物理メモリーのサイズが最小のノードにも適切な値を指定してください。

また、NUMBER IS パラメータの指定には、RMU Open コマンドの Global_Buffers=Total=n 修飾子も使用できます。

```
$ RMU/OPEN/GLOBAL_BUFFERS=(TOTAL=n) mf_personnel
$ rmu -open -global_buffers=¥(total=n¥) mf_personnel
```

RMU Open Global_Buffers=(Total=n) コマンドで指定するのは、このコマンドを発行するノード上にあるデータベースで使用するグローバル・バッファの数の合計です。大量の物理メモリーを搭載したノードでは、物理メモリーのサイズが小さいノードとクラスタを構成した場合、グローバル・バッファの数を増やすことができます。メモリー・サイズが大きい場合にグローバル・バッファを増やすと、ノードのパフォーマンスを向上できます。RMU Open Global_Buffers=(Total=n) コマンドを発行するノードでは、データベースがオープンしている間、グローバル・バッファ・プールには、n で指定した数のグローバル・バッファが存在します。

RMU Open Global_Buffers=(Total=n) コマンドを使用すると、Oracle Rdb は、データベースをオープンするたびにバッファ・プールの再マッピングを実行する必要がなくなるという利点があります。RMU Open コマンドの詳細は、『Oracle RMU Reference Manual』を参照してください。

Performance Monitor の「Buffer Information」画面と SQL SHOW DATABASE 出力は、データベース・ルート・ファイルに格納されているグローバル・バッファ・パラメータを表示し、RMU Show Users コマンドは、グローバル・バッファ・パラメータのアクティブな値を表示します（RMU Open Global_Buffers コマンドでノード上のデータベースをオープンすると、アクティブなパラメータは、ルート・ファイルに格納されている値と異なる場合があります）。例 4-3（4-36）の「Buffer Information」画面では、ルート・ファイルに格納されている NUMBER IS パラメータの値は 250（グローバル・バッファ）ですが、例 4-4（4-37）の RMU Show Users コマンドでは、現行ノードのデータベースでアクティブな NUMBER IS パラメータの値は 350（グローバル・バッファ）となっています。

例 4-3 「Buffer Information」画面でのグローバル・バッファの数

```
Node: TRIXIE          Oracle Rdb V7.0-00 Performance Monitor 21-JUN-1996 11:59:07
Rate: 3.00 Seconds              Buffer Information            Elapsed: 00:00:07.47
Page: 1 of 1       RDBVMS_USER1:[LOGAN.V70]MF_PERSONNEL.RDB;1       Mode: Online
-------------------------------------------------------------------------------
Default user buffer count is 20
Default recovery buffer count is 20
Buffer size is 6 blocks
Global Buffers are disabled
- Global buffer count is 250              <------------ Notice
- Maximum global buffer count per user is 5
- Page transfer via memory is disabled
Global section size with global buffers disabled is 87600 bytes
```

```
- With global buffers enabled is 997430 bytes
Asynchronous pre-fetch is enabled
- Maximum pre-fetch depth is 5 buffers
Detected asynchronous pre-fetch is enabled
- Maximum pre-fetch depth is 4 buffers
- Pre-fetch threshold is 4 pages
Asynchronous batch-write is enabled
- Clean buffer count is 5
- Maximum batch size is 4 buffers
--------------------------------------------------------------------------------
Exit Help Menu Options Refresh Set_rate Write !
```

例 4-4 RMU Show Users コマンドで表示される NUMBER IS パラメータのアクティブな値

```
$ RMU/OPEN/GLOBAL_BUFFERS=(TOTAL=350) mf_personnel
$ RMU/SHOW USERS mf_personnel
Oracle Rdb V7.0-00 on node TRIXIE 28-MAY-1996 10:08:21.49
database _$111$DUA176:[LOGAN.V70]MF_PERSONNEL.RDB;1
    - First opened 28-MAY-1996 10:02:47.44
    * database is opened by an operator
    - current after-image journal file is
_$111$DUA176:[LOGAN.V70]RICK1.AIJ;1
    - global buffer count is 350          <------------ Notice
    - maximum global buffer count per user is 25
    - 325 global buffers free
    - 1 active database user
    - 2080932F:1 - RICK - non-utility, RICK - active user
        - image $111$DUA600:[SQL_X07001.VAX.][CODE]SQL$701.EXE;1
        - 25 global buffers allocated
$
```

NUMBER IS パラメータのデフォルト値は、ユーザーの最大数の 5 倍です。この値は、ユーザー数、データベースのニーズ、使用可能な物理メモリーを基準として決定してください。4.1.2.9 項（4-42）では、グローバル・バッファの数を決める方法について説明します。

システム上で使用可能なメモリーを超えるサイズをグローバル・バッファに定義すると、ページ・フォルトが発生するので、このような定義はしないでください。マルチノード・システムでは、RMU Open コマンドを使用することにより、異なるノードで使用可能なリソースに合わせて NUMBER IS や USER LIMIT パラメータを指定できますが、メモリー容量が最小のノードが制限因子となります。

すべてのユーザーに割り当てられるバッファの合計は、NUMBER IS パラメータで指定したグローバル・バッファの合計以下でなければなりません。これによって、ユーザーが新しいバッファをバッファ・プールに読み込むときに必要となる空きバッファを確保できます。ユーザーが必要なバッファ・サイズの合計が、グローバル・バッファ・プールのサイズを越える場合（データベースへの接続は拒否されます）、NUMBER IS パラメータの値を大きくしてください。必要なサイズが物理メモリーのサイズを超える場合、システムにメモリーを増設するか、マルチノードのノード間でユーザーを分散する必要があります。このような場

合、物理メモリーのサイズより大きな値を NUMBER IS に定義して、ページ・フォルトを発生させることも可能です。ページ・フォルトはパフォーマンスを低下させますが、この場合には許容されます。

4.1.2.8 USER LIMIT パラメータ

USER LIMIT パラメータは、プロセスに割当て可能なグローバル・バッファの最大数です。グローバル・バッファを無効にすると、グローバル・バッファは使用されないので、USER LIMIT パラメータで指定した値は適用されません。

データベースに対してグローバル・バッファを有効にした場合、USER LIMIT パラメータには、NUMBER OF BUFFERS パラメータよりも大きな値に設定してください。USER LIMIT の値が NUMBER OF BUFFERS の値よりも小さいと、ローカル・バッファを有効にした場合よりも、グローバル・バッファを有効にした状態で割り当てられるバッファの数が少なくなります。

USER LIMIT パラメータを設定するには、SQL ALTER DATABASE 文と CREATE DATABASE 文で USER LIMIT IS 句を指定してください。

```
SQL> ALTER DATABASE FILENAME mf_personnel
  1>    GLOBAL BUFFERS ARE ENABLED
  2>    (USER LIMIT IS t);
```

また、USER LIMIT パラメータの指定には、RMU Restore コマンドの Global_Buffers=(User_Limit=t) 修飾子も使用できます。

```
$ RMU/RESTORE/GLOBAL_BUFFERS=(USER_LIMIT=t)/NOCDD mf_pers_backup
$ rmu -restore -global_buffers=¥(user_limit=t¥)  mf_pers_backup
```

ALTER DATABASE 文と CREATE DATABASE 文または RMU Restore コマンドを使用して USER LIMIT パラメータを指定すると、設定値はデータベース・ルート・ファイルに格納されます。この文やコマンドでは、クラスタ内で物理メモリーのサイズが最小のノードにも適切な値を指定してください。

また、USER LIMIT パラメータの指定には、RMU Open コマンドの Global_Buffers=(User_Limit=t) 修飾子も使用できます。

```
$ RMU/OPEN/GLOBAL_BUFFERS=(USER_LIMIT=t) mf_personnel
$ rmu -open -global_buffers=¥(user_limit=t¥) mf_personnel
```

RMU Open Global_Buffers=(User_Limit=t) コマンドで指定するのは、このコマンドを発行するノード上にあるデータベースで使用するグローバル・バッファの最大数です。大量の物理メモリーを搭載したノードでは、物理メモリーのサイズが小さいノードとクラスタを構成した場合、グローバル・バッファの数を増やすことができます。メモリー・サイズが大きい場合に、プロセスあたりのグローバル・バッファを増やすと、ノードのパフォーマンスを向上できます。RMU Open Global_Buffers=(User_Limit=t) コマンドを発行するノードでは、データベースがオープンしている間にプロセスへ割当て可能なグローバル・バッファの最大数は、t で指定された値となります（データベースをクローズするまでの間、RMU Open

Global_Buffers コマンドで指定した USER LIMIT の値は、データベース・ルート・ファイルに格納されている USER LIMIT の値よりも優先されます)。データベースが既にオープンしている場合、RMU Open Global_Buffers=(User_Limit=t) コマンドは失敗します。RMU Open コマンドの詳細は、『Oracle RMU Reference Manual』を参照してください。

「Buffer Information」画面と SQL SHOW DATABASE 出力は、データベース・ルート・ファイルに格納されているグローバル・バッファ・パラメータを表示し、RMU Show Users コマンドは、グローバル・バッファ・パラメータのアクティブな値を表示します（RMU Open Global_Buffers コマンドでノード上のデータベースをオープンすると、アクティブなパラメータは、ルート・ファイルに格納されている値と異なる場合があります)。例 4-5（4-39）の「Buffer Information」画面では、ルート・ファイルに格納されている USER LIMIT パラメータの値は 25（グローバル・バッファ）ですが、例 4-6（4-39）の RMU Show Users コマンドでは、現行ノードのデータベースでアクティブな USER LIMIT パラメータの値は 50（グローバル・バッファ）となっています。

例 4-5 Performance Monitor の「Buffer Information」画面で表示される USER LIMIT パラメータのアクティブな値

```
Node: TRIXIE          Oracle Rdb V7.0-00 Performance Monitor 21-JUN-1996 11:59:07
Rate: 3.00 Seconds              Buffer Information            Elapsed: 00:00:07.47
Page: 1 of 1        RDBVMS_USER1:[LOGAN.V70]MF_PERSONNEL.RDB;1       Mode: Online
--------------------------------------------------------------------------------
Default user buffer count is 20
Default recovery buffer count is 20
Buffer size is 6 blocks
Global Buffers are disabled
- Global buffer count is 250
- Maximum global buffer count per user is 25  <------------ Notice
- Page transfer via memory is disabled
Global section size with global buffers disabled is 87600 bytes
- With global buffers enabled is 997430 bytes
Asynchronous pre-fetch is enabled
- Maximum pre-fetch depth is 5 buffers
Detected asynchronous pre-fetch is enabled
- Maximum pre-fetch depth is 4 buffers
- Pre-fetch threshold is 4 pages
Asynchronous batch-write is enabled
- Clean buffer count is 5
- Maximum batch size is 4 buffers
--------------------------------------------------------------------------------
Exit Help Menu Options Refresh Set_rate Write !
```

例 4-6 RMU Show Users コマンドで表示される USER LIMIT パラメータのアクティブな値

```
$ RMU/OPEN/GLOBAL_BUFFERS=(USER_LIMIT=50) mf_personnel
Oracle Rdb V7.0-00 on node TRIXIE 28-MAY-1996 10:21:08.13
database _$111$DUA176:[LOGAN.V70]MF_PERSONNEL.RDB;1
```

```
    - First opened 28-MAY-1996 10:20:53.50
    * database is opened by an operator
    - current after-image journal file is
_$111$DUA176:[LOGAN.V70]RICK1.AIJ;1
    - global buffer count is 250
    - maximum global buffer count per user is 50  <------------ Notice
    - 200 global buffers free
    - 1 active database user
    - 2080932F:1 - RICK - non-utility, RICK - active user
        - image $111$DUA600:[SQL_X07001.VAX.][CODE]SQL$701.EXE;1
        - 50 global buffers allocated
```

データベースに対してグローバル・バッファを有効にすると、プロセスは、RDM$BIND_BUFFERS 論理名または RDB_BIND_BUFFERS 構成パラメータで、データベースへの接続時にプロセスに割り当てられるグローバル・バッファの数を定義できます。RDM$BIND_BUFFERS または RDB_BIND_BUFFERS には、有効な値として、正の整数を指定してください。RDM$BIND_BUFFERS または RDB_BIND_BUFFERS で指定した数が、USER LIMIT パラメータのアクティブな値よりも小さい場合、RDM$BIND_BUFFERS または RDB_BIND_BUFFERS で指定した数のグローバル・バッファが割り当てられます。例 4-7 (4-40) を参照してください。

例 4-7 RDM$BIND_BUFFERS 論理名を使用して、アクティブな USER LIMIT パラメータよりも小さな値をグローバル・バッファの数に指定

```
$ ! A user executes the following commands:
$ DEFINE RDM$BIND_BUFFERS 8
$ !
$ SQL
SQL> ATTACH 'FILENAME mf_personnel';
.
.
.
$!
$! Another user on the same node as the first user executes
$! the following command:
$ RMU/SHOW USERS mf_personnel
Oracle Rdb V7.0-00 on node TRIXIE 28-MAY-1996 10:27:59.10
database _$111$DUA176:[LOGAN.V70]MF_PERSONNEL.RDB;1
    - First opened 28-MAY-1996 10:20:53.50
    * database is opened by an operator
    - current after-image journal file is
_$111$DUA176:[LOGAN.V70]RICK1.AIJ;1
    - global buffer count is 250
    - maximum global buffer count per user is 25
    - 242 global buffers free
    - 1 active database user
    - 2080F11A:1 - RICK - non-utility, RICK - active user
```

```
        - image $111$DUA600:[SQL_X07001.VAX.][CODE]SQL$701.EXE;1
        - 8 global buffers allocated
$
```

RDM$BIND_BUFFERS 論理名または RDB_BIND_BUFFERS 構成パラメータで指定した数が、USER LIMIT パラメータのアクティブな値よりも大きい場合、プロセスには、USER LIMIT で指定した数のグローバル・バッファが割り当てられます。例 4-8（4-41）を参照してください。

例 4-8 RDM$BIND_BUFFERS 論理名を使用して、アクティブな USER LIMIT パラメータよりも大きな値をグローバル・バッファの数に指定

```
$! A user executes the following commands:
$ DEFINE RDM$BIND_BUFFERS 50
$!
$ SQL
SQL> ATTACH 'FILENAME mf_personnel';
.
.
.
$!
$! Another user on the same node as the first user executes
$! the following command:
$ RMU/SHOW USERS mf_personnel
Oracle Rdb V7.0-00 on node TRIXIE 28-MAY-1996 10:39:00.12
database _$111$DUA176:[LOGAN.V70]MF_PERSONNEL.RDB;1
    - First opened 28-MAY-1996 10:20:53.50
    * database is opened by an operator
    - current after-image journal file is
_$111$DUA176:[LOGAN.V70]RICK1.AIJ;1
    - global buffer count is 250
    - maximum global buffer count per user is 25
    - 225 global buffers free
    - 1 active database user
    - 20808D50:1 - _RTA7: - non-utility, RICK - active user
        - image $111$DUA600:[SQL_X07001.VAX.][CODE]SQL$701.EXE;1
        - 25 global buffers allocated
$
```

注意：ユーザーは、グローバル・バッファ・プール内のすべてのバッファをいつでも使用できます。バッファの割当てとバッファ・リミットは、プロセスが一度に使用できるバッファの数を示します。たとえば、グローバル・バッファ・プールに 1000 個のバッファがあり、100 人のユーザーにそれぞれ 10 個のバッファを割り当てる場合、各ユーザーは、ディスク I/O を実行しなくても 1000 個の中から任意のバッファを読み取ることができますが、一度に読取りまたは変更できるのは 10 個のみです。ユーザーが一度に読取りまたは変更できる 10 個のバッファは、ユーザーの割当てセットと呼ばれます。

各プロセスが割り当てることができるグローバル・バッファの最大数を計算するには、NUMBER IS 修飾子で指定したグローバル・バッファの最大数を、データベース・アクセスを保証したいプロセスの合計で割ります。たとえば、グローバル・バッファの合計が 200 で、少なくとも 10 人のユーザーによるデータベース・アクセスを保証したい場合、各プロセスのグローバル・バッファの最大数を 20 に設定します（USER LIMIT は 20）。4.1.2.9 項（4-42）では、プロセスに割当て可能なグローバル・バッファの数を決める方法について説明します。NUMBER OF BUFFERS を 10 に設定することもでき、この設定では 20 人のユーザーがデータベースに接続して通常の処理を実行できます。ただし、USER LIMIT を 20 に設定すると、デフォルト値（RDM$BIND_BUFFERS 論理名または RDB_BIND_BUFFERS 構成パラメータを使用）よりも大きな数のバッファを要求する特殊なアプリケーションが有効になり、最大 20 のバッファが割当て可能になります。

USER LIMIT パラメータには、グローバル・バッファの合計よりも大きな値を指定しないでください。デフォルト値は、5 です。

USER LIMIT パラメータは、システムの処理とパフォーマンスに重要な影響を与えます。大きな値を指定すると、データベース・アクセスを保証するプロセスの数は少なくなります。USER LIMIT パラメータで指定した値は、次の値に関係なく、プロセスによる割当てが可能なバッファの最大数を決定します。

- プロセスが使用するローカル・バッファのデフォルト数。SQL CREATE DATABASE 文の NUMBER OF BUFFERS パラメータで指定します。
- RDM$BIND_BUFFERS または RDB_BIND_BUFFERS で定義したバッファの数。

4.1.2.9 グローバル・バッファのチューニング

ノード上のデータベース・グローバル・バッファ・プール用に予約されている仮想メモリーの容量は、BUFFER SIZE パラメータの値と NUMBER IS の値を掛けて計算します。

バッファ・プールのサイズは、システム・パフォーマンスに重大な影響を与えます。

- バッファ・プールが大きすぎると、オペレーティング・システムは、強制的に仮想ページングを実行します。この場合、データベース・ページをメモリーに読み込むためには、

1 回目の I/O で、オペレーティング・システムがバッファをワーキング・セットにページ・フォルトし、2 回目の I/O で、Oracle Rdb がページをバッファ・プールに読み込むので、合計 2 回の I/O を実行しなければなりません。

OpenVMS VAX OpenVMS Alpha

バッファ・プールが大きすぎる可能性がある場合は、SHOW SYSTEM コマンドを実行して、システムがどの程度のページングを実行しているか調べてください。◆

- バッファ・プールが小さすぎると、Oracle Rdb が実行するデータベース I/O の回数が多くなります。Performance Monitor の「PIO Statistics - Data Writes」画面、「PIO Statistics - Data Fetches」画面、「PIO Statistics - SPAM Fetches」画面を使用して、過剰な I/O が発生していないか確認してください。各画面の詳細は、4.1.1.2 項（4-8）、4.1.1.3 項（4-9）、4.1.1.4 項（4-10）を参照してください。

ここでは、NUMBER IS、USER LIMIT および NUMBER OF BUFFERS パラメータの値を設定する方法を説明します。マルチノード・システムで最大限のパフォーマンスを発揮するには、クラスタを構成する各ノードの NUMBER IS および USER LIMIT グローバル・バッファ・パラメータの値を、RMU Open Global_Buffers=(Total=n,User_Limit=t) コマンドを使用してチューニングしてください。

注意：グローバル・バッファ・パラメータのデフォルト値を使用すると、データベースで最適なパフォーマンスを発揮することは期待できません。多種多様なデータベースやシステム構成に対応できるデフォルト値を提供するのは不可能です。

- NUMBER IS
  グローバル・バッファの合計は、次の条件を考慮して設定します。
  - 使用可能な物理メモリーの容量
  - Oracle Rdb のデータベースに割当て可能な物理メモリーのパーセンテージ

  たとえば、使用可能な物理メモリーは 128MB、データベースに割り当てられるのは 10%、バッファ・サイズは 3KB だとすると、次のような式になります。
  NUMBER IS = (128 × 0.10) / 0 .003 = 12 / 0 .003 = 4000
  このシステムは、4000 個のバッファで構成されるグローバル・バッファ・プールをサポートできます。
- USER LIMIT
  この値は、データベース・アクセスを保証したいユーザーの人数を基準に設定します。同じ例を使うと、50 人のユーザーのアクセスを保証したい場合、次のような式になります。
  USER LIMIT = 4000 / 50 = 80
  このシステムは、最低 50 人のユーザーによるデータベース・アクセスを保証し、各ユーザーに 80 個のバッファを割り当てることができます。

- NUMBER OF BUFFERS
  このパラメータの値は、データベース処理がピーク時の間、アクセスが必要なユーザー数に基づいて設定します。同じ例を使うと、ピーク時に 200 人のユーザーがアクセスする場合、次のような式になります。
  NUMBER OF BUFFERS = 4000 / 200 = 20
  このシステムでは、デフォルトとして 20 個のバッファを 200 人のユーザーに割り当てることができます。

4.1.2.10 グローバル・バッファ・オーバーフロー管理の利点

同時ユーザーがデータを共有する環境でグローバル・バッファを有効にすることの大きな利点は、特定ノード上でデータベース全体にわたって定義されたすべてのグローバル・バッファに対して、ユーザーが読取りアクセスできることです。ユーザーが新しいページから 1 行読み取る場合、そのノード上のデータベースのプールにあるグローバル・バッファ内にページが存在し、バッファが共有されていれば、ディスク I/O を実行する必要はありません。グローバル・バッファ・プールではページを共有できるので、ディスク I/O を減らすことができます。必要なバッファの数も少なくなるので、使用するメモリーも少なくなります。

データベース・プロセスは、割当てセットにあるグローバル・バッファ数の書込みや制御ができます。ユーザーがページを読み取るとき、そのページを格納しているグローバル・バッファは、ユーザーの割当てセットの一部になります。

ユーザーによる書込みや制御が可能なグローバル・バッファの最大数は、USER LIMIT パラメータのアクティブ値で指定します。このパラメータは、接続時に、Oracle Rdb Monitor がデータベース・ユーザーに割り当てることのできるグローバル・バッファの最大数を示します。ユーザーに割り当てられるグローバル・バッファの数のデフォルト値は、NUMBER OF BUFFERS パラメータで指定します。プロセスに実際に割り当てられるグローバル・バッファの数は変動し、RDM$BIND_BUFFERS 論理名または RDB_BIND_BUFFERS 構成パラメータを使用して、ユーザーごとに設定できます。RDM$BIND_BUFFERS または RDB_BIND_BUFFERS を定義した場合、この値がアクティブな USER LIMIT の値以下であれば、デフォルト値よりも優先します。

データベース接続が成功するには、接続時に、必要な数のバッファが、データベース・グローバル・バッファ内に割当て可能な状態で存在しなければなりません。割り当てるグローバル・バッファが足りないと、接続は失敗します。

これまでに説明したとおり、データベース・プロセスは、割当てセットにあるグローバル・バッファの書込みや制御ができます。ただし、ユーザーの割当てセットに割り当てられていないグローバル・バッファが存在する場合、割当てセットのサイズを超えるデータを、グローバル・バッファに格納することがあります。

このような処理が実行されるのは、次に説明するように、バッファがオーバーフローした場合、LRU バッファ置換アルゴリズムの改良バージョンを使用して、置換対象となるバッファを選択するからです。ユーザーに割り当てられているバッファの総数が、データベースで定義されているグローバル・バッファの総数よりも少ない場合に発生します。

たとえば、データベース全体を対象に定義されたグローバル・バッファの合計が 2,000 で、USER LIMIT に 100 個のバッファを指定しているとします（最大割当てセットでは、最大 20 の同時接続が可能）。10 人のユーザーが最大数の割当てセットに同時接続する場合、ユーザーに割り当てられるグローバル・バッファはわずか 50%、つまり 1,000 個であり、残りの 1,000 個は未使用となります。ただし、ユーザーによるアクセス可能なデータが格納されているバッファの数は、割当て済である 1,000 個をすぐに超えてしまいます。

ユーザーのバッファ・プールがいっぱいの状態で、別のページが要求された場合、Oracel Rdb は、LRU アルゴリズムを使用して、新しいページを格納するために置換対象となるバッファを選択します。プール・オーバーフローの処理は、グローバル・バッファとローカル・バッファで異なります。

ローカル・バッファでは、置換対象となるバッファに格納されているデータは、グローバル・バッファから削除され、新しいバッファに置き換えられます。置換対象となったバッファに格納されていたデータの読取り要求が発生したら、データをディスクから再度読み込みます。

グローバル・バッファでは、グローバル・バッファの割当てセットがいっぱいの状態で、新しいページをユーザーの割当てセットに読み込む必要がある場合、データを置き換えた後も、置換対象となったバッファがメモリー内にとどまるという利点があります。目的のページが含まれる新しいバッファを読み込めるように、ユーザーの割当てセット内に空きを作るため、LRU バッファ置換アルゴリズムを使用して割当てセットの中から置換対象のバッファを選択しますが、グローバル・バッファでは、置換対象となったバッファはグローバル・バッファ・プールから削除されるとは限りません。

新しいバッファの格納先として、まずグローバル・バッファ・プール内で、割り当てられていない空きバッファが選択され、次に最も古いデータがいっぱいに格納された、未割当てバッファが選択されます。Oracle Rdb は、擬似 LRU アルゴリズムを使用して、グローバル・バッファ・プールのバッファを選択します。データベース・グローバル・バッファ・プールに割り当てられていないバッファが存在する場合、新しいバッファとして使用されます。この場合、ユーザーの内部グローバル・バッファ制御構造は更新され、メモリー内の新しいバッファのロケーションをポイントしますが、ユーザー構造で前にポイントしていた置換対象のバッファは、グローバル・メモリー内にとどまります。ユーザーが、置換対象のバッファにあったデータ（ユーザーの割当てセットには現在含まれていない）を再度読み取る場合、データベースのグローバル・バッファ・プール内にあるので、ディスクからデータを読み込む必要はありません。

ただし、すべてのグローバル・バッファがユーザーに割り当てられている場合、置換対象のバッファは新しいバッファに置き換えられるので、バッファ内のデータは、ユーザーの割当てセットだけでなく、グローバル・バッファ・プールからも削除されます。したがって、置換の対象となったバッファに格納されていたデータの読取り要求が発生したら、データをディスクから再度読み込みます。

シングル・ユーザー環境では、Oracle Rdb が実行するグローバル・バッファ管理やユーザー・バッファ・オーバーフロー管理により、ユーザーは、データベースに割り当てられたすべての

グローバル・バッファに効率的な書込みを実行できます。マルチユーザー環境では、ユーザーが所有および制御できるのは割当てセット内のバッファのみなので、新しいバッファを格納できる未割当てのバッファが存在する可能性は低いといえます。ただし、目的のバッファが既にメモリーに読み込まれている可能性があります（他のユーザーによる読取り）。

グローバル・バッファ・プールにデータベース全体が収まるが、ユーザーの割当てセットのサイズを超えてしまう場合でも、上記のような利点があるため、データベース全体をメモリーに格納できます。データベースをグローバル・バッファに読み込んでしまえば、その後の読取りはメモリーから実行できます。読取り / 書込み環境では、実行しなければならないのはディスクへの書込みのみです。このように、データベース全体をメモリーに格納することには大きな利点があり、Oracle Rdb グローバル・バッファの重要な機能です。同様に、頻繁にアクセスされるデータベースのサブセットをグローバル・バッファに格納する場合にも、同じような利点があります。製造設備などがこの例で、履歴データはオンラインで保存されますが、ユーザーは現在のデータにしかアクセスしない環境で利点があります。

例 4-9（4-46）は、グローバル・バッファにデータベースを格納した場合、読取り操作の回数がどの程度減少するかを示しています。例 4-9（4-46）は、ページ・サイズはバッファ・サイズと等しく、2 人のユーザーはデータベースへ同時に接続し、これまでにページ・フェッチは発生していないということを前提としています。簡易化するために、メタデータ・バッファは、プールから除外します。

例 4-9（4-46）では、次のグローバル・バッファ・パラメータが有効になっています。

- グローバル・バッファを有効化
- グローバル・バッファ・カウント（NUMBER IS パラメータ）は 6
- ユーザーあたりのグローバル・バッファ・カウントの最大値（USER LIMIT パラメータ）は 3
- ユーザーあたりのデフォルト・バッファ（NUMBER OF BUFFERS パラメータ）は 2
- ユーザー 1 とユーザー 2 の 2 人のユーザーが同じノード N1 に存在
- ユーザー 2 は、RDM$BIND_BUFFERS または RDB_BIND_BUFFERS を 3 に定義
- ユーザー 1 はバッファを 2 個、ユーザー 2 はバッファを 3 個使用

この例では、7 つの手順が発生し、処理内容を説明しています。

例 4-9 グローバル・バッファを有効化した場合の読取り操作の減少

```
Step  User 1   User 2         Global Buffer Page Walk-Through
      Page     Page                    Comments
      Fetch    Fetch
-------------------------------------------------------------------
1       1              User 1 reads page 1.
2       2              User 1 reads page 2.
3       3              Page 1 is the victim, User 1 reads page 3
```

```
                        into a new, unallocated buffer.  Page 1
                        stays in the global buffer pool.
4       1               Page 2 is the victim, User 1 finds page 1
                        in the global buffer pool, saving a read
                        operation.  Page 2 stays in the global
                        buffer pool.
5               1       User 2 finds page 1 in the global buffer
                        pool, saving a read operation.
6       2               Page 3 is the victim and remains in the
                        global buffer pool.  User 1 finds page 2
                        in the global buffer pool and saves a
                        read operation.
7               3       User 2 finds page 3 in the global buffer
                        pool and saves a read operation.
```

例 4-9（4-46）は、マルチユーザー環境において、ユーザーがデータを共有する場合、グローバル・バッファによってディスク I/O が減少する仕組みと、グローバル・バッファで置換対象バッファを管理することにより、さらにディスク I/O を減少させる方法を示しています。

この例では、Oracle Rdb は、置換対象バッファの処理により、ユーザーの割当てセットを超えるサイズのデータをグローバル・バッファに格納しています。データベースのグローバル・バッファのサイズを増やすことにより、要求されたメモリーがグローバル・バッファ・プールに存在する確率が増えるので、ダイレクト I/O が減少します。データベース全体がグローバル・バッファ・プールに格納されるまでは、グローバル・バッファ・プールから削除されるデータもあるので、必要なページをプールに戻すためにダイレクト I/O が発生します。

ただし、割当てセットのサイズを小さくすることを推奨しているわけではありません。ユーザー割当てセットのサイズは、アプリケーションのパフォーマンスやバッファ・キャッシュの効率に大きな影響を与え、特にマルチユーザー環境では重要です。読取り集中型の環境では、キャッシュの有効性が高ければ、目的のデータをディスクから読み込む回数が減少します。更新集中型の環境では、バッチ書込み操作において、バッファがオーバーフローすると、マークした（更新した）バッファをディスクへフラッシュするので、ユーザーのバッファ・プールのサイズは、ディスク書込みの回数に影響します。割当てセットのサイズが小さいと、ディスクへのバッチ書込み操作が頻繁に発生し、バッチ書込みの最適化による効果がなくなります。

4.1.2.11 グローバル・バッファ・メモリー内にデータを保持する利点

Oracle Rdb データベースを RMU Open コマンドで明示的にオープン（オープンした状態にする）すると、データベース・アクセスによって Oracle Rdb グローバル・バッファに格納されたデータは、データベース接続を通じて、メモリー内に残ります。これにより、複数のデータベース接続でデータへのアクセスが発生すれば、I/O を大幅に減少できます。たとえば、最初にアプリケーション・プログラムを実行し、データの読込みとバッファのロードを実行しておけば、必要なデータベース・ページの多く（すべてではないとしても）は、メモリー内に存在します。したがって、最初にアプリケーションを実行した後、実行を続けるた

びに、グローバル・バッファには必要なデータが既に読み込まれているため、Performance Monitor が示すデータ・ファイル読取り回数は減少していきます。

この利点を活かすためには、RMU Open コマンドでデータベースを明示的にオープン（その後オープンした状態にする）してください。複数のデータベース接続でグローバル・バッファ・メモリー内にデータが保持されるため、アプリケーションの起動や応答時間も高速化します。

4.1.2.12 グローバル・バッファを有効化したときのパラメータ変更

データベースに対してローカル・バッファを有効にすると、Oracle Rdb Monitor は、最初のデータベース接続で、グローバル・セクションまたは共有メモリー・パーティションを作成し、データベース・ルート構造を格納します。データベースに対してグローバル・バッファを有効にすると、グローバル・セクションまたは共有メモリー・パーティションを拡張し、格納されているデータの整合性を確保するために必要となるグローバル・バッファ・プールとデータ構造を格納します。このデータ構造を使用して、次のような情報を保持します。

- グローバル・バッファ制御ブロック
- 割当てセット・ブロック件数リスト
- システム所有のデータベースのプロセス・ローカル・ロックと記憶領域ロック

このように、グローバル・バッファでは追加のデータ構造を使用するので、データベースに対してグローバル・バッファを有効化するときは、既存のシステム・パラメータを一部変更する必要があります。Performance Monitor の「Buffer Information」画面では、データベースに対してグローバル・バッファを有効化したときと無効化したときの、グローバル・セクションや共有メモリー・パーティションのサイズを表示します。データベースのグローバル・セクションまたは共有メモリー・パーティションのサイズがわかれば、既存のシステム・パラメータを変更するべきかどうかを判断できます。

「Buffer Information」画面で表示されるのは、グローバル・セクションまたは共有メモリー・パーティションのサイズの推定値です。例 4-10（4-48）は、「Buffer Information」画面です。

例 4-10 「Buffer Information」画面でグローバル・セクションのサイズを把握

```
Node: TRIXIE          Oracle Rdb V7.0-00 Performance Monitor 21-JUN-1996 10:36:06
Rate: 3.00 Seconds              Buffer Information           Elapsed: 01:32:15.48
Page: 1 of 1       RDBVMS_USER1:[LOGAN.V70]MF_PERSONNEL.RDB;1       Mode: Online
------------------------------------------------------------------------------
Default user buffer count is 20
Default recovery buffer count is 20
Buffer size is 6 blocks
Global Buffers are enabled
- Global buffer count is 1000
- Maximum global buffer count per user is 40
- Page transfer via memory is disabled
```

```
Global section size with global buffers disabled is 548934 bytes  <----- Notice
- With global buffers enabled is 4078056 bytes                    <-----
Asynchronous pre-fetch is enabled
- Maximum pre-fetch depth is 10 buffers
Detected asynchronous pre-fetch is enabled
- Maximum pre-fetch depth is 4 buffers
- Pre-fetch threshold is 4 pages
Asynchronous batch-write is enabled
- Clean buffer count is 4
- Maximum batch size is 4 buffers
------------------------------------------------------------------------------
Exit Help Menu Options Refresh Set_rate Write !
```

最初の値（グローバル・バッファは無効）は、グローバル・バッファを無効にした場合のデータベースのグローバル・セクションのサイズです。次の値（グローバル・バッファは有効）は、データベースに対してグローバル・バッファを有効にした場合のグローバル・セクションのサイズです。この 2 つの値は、データベース・ルート・ファイルには格納されていませんが、ルート・ファイルに格納されている値を元に計算されています。「Buffer Information」画面を使えば、実際にグローバル・バッファを有効にする前に、データベースに対してグローバル・バッファを有効化した場合に必要なグローバル・セクションのサイズを計算できます。

データベースに対してグローバル・バッファを無効にした状態で、BUFFER SIZE、NUMBER IS、USER LIMIT および NUMBER OF BUFFERS パラメータを、データベースに対してグローバル・バッファを有効化した場合の値に設定します。また、その他のデータベース・パラメータは、グローバル・バッファを有効にしたときにデータベースで使用する値に設定してください。次に、「Buffer Information」画面を表示します。グローバル・バッファを有効にするとグローバル・セクションのサイズが大きくなるので、システム・パラメータやプロセス・クォータの値を大きくする必要があるかどうかを、実際にグローバル・バッファを有効にする前に判断できます。

例 4-10（4-48）の「Buffer Information」画面を使用して、グローバル・バッファが 1000、ユーザーあたりのグローバル・バッファの上限が 40、バッファ・サイズが 6 ブロックのデータベースについて、グローバル・セクションに必要なバイト数を計算します。

グローバル・セクション・サイズの値は、グローバル・バッファを有効にした場合、無効にした場合の 7 倍になっていることを示しています。

オペレーティング・システムによってページ・サイズは異なるため（OpenVMS では 1 ページ 512 バイトですが、他のオペレーティング・システムとは異なる可能性があります）、グローバル・セクションのサイズは、ページ単位ではなくバイト数で表示されます。データベースのグローバル・セクションのページ数は、次の式で計算できます。

```
derived number of bytes / bytes per page = number of pages for global section
```

次に、例 4-10（4-48）の mf_personnel データベースでグローバル・バッファを有効にした場合、グローバル・セクションのサイズからグローバル・セクションのページ数を計算します。

```
4078056 / 512 = 7964.9
```

ここで計算したグローバル・セクションまたは共有メモリー・パーティションのページ数は、最も近い整数に切り上げてください。この例では、グローバル・バッファを有効にした場合のグローバル・セクションは 7965 ページになります。また、グローバル・セクションに十分なページを割り当てるために、計算結果に 10 ～ 15 ページを追加してください。この例では、データベースに対してグローバル・バッファを有効にした場合に必要となるグローバル・セクションは、7980 ページです。

グローバル・バッファを有効にするには、データベースをオフラインにしてください。

OpenVMS VAX OpenVMS Alpha

OpenVMS SYSGEN パラメータの変更には、システムをシャットダウンし、リブートする必要があります。SYSGEN パラメータの変更には、MODPARAMS.DAT ファイルと OpenVMS AUTOGEN 機能を使用することをお薦めします。パラメータをハードコードすると、システムのリブートでハングする可能性があります。

また、SYSGEN パラメータを変更すると、ページング・ファイル・クォータが正しいことを保証するために、Oracle Rdb Monitor プロセスのアカウント・クォータの変更が必要になる場合があります。詳細は、この項の GBLPAGFIL パラメータの説明を参照してください。

また、この項では、データベースに対してグローバル・バッファを有効にした場合の GBLSECTIONS、GBLPAGES、GBLPAGFIL、VIRTUALPAGECNT および PGFLQUOTA パラメータの設定について説明します。

GBLSECTIONS パラメータ

GBLSECTIONS パラメータは、システム上で作成可能なグローバル・セクションの総数です。グローバル・セクションは、グローバル・バッファの有効化と無効化に関係なく、データベースを最初にオープンしたときに、データベース・ルートに 1 つ割り当てられます。グローバル・バッファの有効化と無効化に関係なく、データベースは 1 つのグローバル・セクションを使用するため、グローバル・バッファを有効にしても、GBLSECTIONS パラメータを変更する必要はありません。

GBLPAGES パラメータ

GBLPAGES パラメータは、システム上に作成可能なグローバル・ページの総数です。GBLPAGES と VIRTUALPAGECNT は、最も重要なパラメータです。この項では、VIRTUALPAGECNT パラメータを詳細に説明しているので参照してください。

GBLSECTIONS、GBLPAGES、GBLPAGFIL および VIRTUALPAGECNT パラメータは、変更できますが、動的ではないため、変更後にシステムをリブートする必要があります。

GBLPAGES と GBLSECTIONS パラメータは、OpenVMS System Generation ユーティリティ（SYSGEN）、OpenVMS Install ユーティリティ（INSTALL）、字句関数で確認できます。SYSGEN SHOW GBLPAGES および SHOW GBLSECTIONS コマンドを実行すると、GBLPAGES および GBLSECTIONS エントリの現在の値が表示されます。

```
$ MCR SYSGEN
SYSGEN>  SHOW GBLPAGES
Parameter Name            Current    Default     Min.     Max.     Unit  Dynamic
--------------            -------    -------    -------  -------   ----  -------
GBLPAGES                   350000      10000       512        -1 Pages
SYSGEN>  SHOW GBLSECTIONS
Parameter Name            Current    Default     Min.     Max.     Unit  Dynamic
--------------            -------    -------    -------  -------   ----  -------
GBLSECTIONS                  2500        250        60      4095 Sections
```

GBLSECTIONS エントリの現在の値は 2500、GBLPAGES エントリは 350000 です。

INSTALL プロンプトで LIST/GLOBAL/SUMMARY コマンドを発行すると、使用中のグローバル・セクションとグローバル・ページの数がわかります。

```
$ INSTALL
INSTALL> LIST/GLOBAL/SUMMARY
        Summary of Local Memory Global Sections
    1022 Global Sections Used,  132790/217210 Global Pages Used/Unused
```

SYSGEN が表示した 2500 の合計のうち、1022 のグローバル・セクションが使用中なので、使用可能なグローバル・セクションは 1478 です。SYSGEN が表示した 350000 の合計のうち、132790 のグローバル・ページが使用中なので、使用可能なグローバル・ページは 217210 です。

字句関数 F$GETSYI を使用しても、GBLPAGES および GBLSECTIONS エントリの使用可能な値を表示できます。次のシンボルを定義して DCL レベルまたは対話型 SQL で起動すると、GBLPAGES および GBLSECTIONS エントリの使用可能な値が表示されます。

```
$ GBLPAGES    :==       "write sys$output f$getsyi(""""free_gblpages"""")
$ GBLSECTIONS :==       "write sys$output f$getsyi(""""free_gblsects"""")
```

シンボルが定義されていれば、使用可能な GBLPAGES と GBLSECTIONS の数を確認できます。

```
$ GBLPAGES
217210
$ GBLSECTIONS
1478
```

字句関数 F$GETSYI が返す値は、INSTALL/LIST/GLOBAL/SUMMARY コマンドが表示するグローバル・ページとグローバル・セクションの値と同じであることに注意してください。

グローバル・バッファを有効化するようデータベースを変更する前に、ローカル・バッファを有効化して、簡単なデータベース接続を実行してください。これにより、グローバル・セクションが割り当てられ、データベース・ルート構造に格納されているグローバル・ページの数がわかります。

```
$ SQL
SQL> ATTACH 'FILENAME mf_personnel';
SQL> SHOW DATABASE *
   .
   .
   .
        Number of users:               50
        Number of nodes:               16
        Buffer Size (blocks/buffer):   6
        Number of Buffers:             20
        Number of Recovery Buffers:    20
        Snapshots are Enabled Immediate
        Carry over locks are enabled
        Lock timeout interval is 0 seconds
        Adjustable lock granularity is enabled
        Global buffers are disabled (number is 250, user limit is 5)
   .
   .
   .
SQL> $ GBLPAGES
217122
SQL> $ GBLSECTIONS
1477
```

"$ GBLPAGES" は、使用可能なグローバル・ページ・エントリの数を示し、217210 から 217122 に減少していることがわかります（88 エントリが割当て済）。"$ GBLSECTIONS" は、使用可能なグローバル・セクション・エントリの数を示し、1478 から 1477 に減少していることがわかります（1 つのグローバル・セクションが割当て済）。データベースをオープンすると、グローバル・セクションが 1 つ割り当てられます。

次の例では、データベースに対してグローバル・バッファを有効にし、バッファ・サイズが 6 ブロックのグローバル・バッファを 1000 個割り当てます。

```
$ SQL
SQL> ALTER DATABASE FILENAME mf_personnel
  1> GLOBAL BUFFERS ARE ENABLED
  2> (NUMBER IS 1000, USER LIMIT 40);
SQL> --
SQL> ATTACH 'FILENAME mf_personnel';
SQL> --
SQL> SHOW DATABASE *
   .
   .
   .
        Number of users:               50
        Number of nodes:               16
        Buffer Size (blocks/buffer):   6
```

```
       Number of Buffers:             20
       Number of Recovery Buffers:    20
       Snapshots are Enabled Immediate
       Carry over locks are enabled
 Lock timeout interval is 0 seconds
 Adjustable lock granularity is enabled
 Global buffers are enabled (number is 1000, user limit is 40)
   .
   .
   .
SQL> --
SQL> $ GBLPAGES
210262
SQL> $ GBLSECTIONS
1477
```

データベースに接続する前、グローバル・ページ・エントリの値は 217210 です（データベースに対してグローバル・バッファを有効にする前）。データベースに接続した後、GBLPAGES の値は 217210 から 210262 に減少しています。つまり、6948 のグローバル・ページがグローバル・セクションに割り当てられています。

最後に、最大ユーザー数を 75 に設定し、データベースに対する After-image ジャーナルを有効にします。

```
SQL> ALTER DATABASE FILENAME mf_personnel
  1>    NUMBER OF USERS IS 75
  2>    JOURNAL FILENAME SQL_DISK1:[RICK]MF_PERS_JOURNAL
  3>    JOURNAL ALLOCATION IS 500 BLOCKS
  4>    JOURNAL EXTENT IS 100 BLOCKS;
%RDMS-W-DOFULLBCK, full database backup should be done to ensure future recovery
SQL> --
SQL> ATTACH 'FILENAME mf_personnel';
SQL> SHOW DATABASE *
   .
   .
   .
       Number of users:               75
       Number of nodes:               16
       Buffer Size (blocks/buffer):   6
       Number of Buffers:             20
       Number of Recovery Buffers:    20
       Snapshots are Enabled Immediate
       Carry over locks are enabled
 Lock timeout interval is 0 seconds
 Adjustable lock granularity is enabled
 Global buffers are enabled (number is 1000, user limit is 40)
   .
   .
```

```
   .
        AIJ File Allocation:          500
        AIJ File Extent:              100
   .
   .
   .
SQL> --
SQL> $ GBLPAGES
209242
SQL> $ GBLSECTIONS
1477
```

GBLPAGES の値は、217210 から 209242 に減少しています。つまり、7968 のグローバル・ページがグローバル・セクションに割り当てられています。この例で使用したデータベースのグローバル・バッファやデータベースの特性などは、例 4-10（4-48）のデータベースと同じです。例 4-10（4-48）では、データベースに対してグローバル・バッファを有効にした場合、「Buffer Information」画面では、グローバル・セクションの見積り値として 7965 が表示されていました。この例では、このデータベースに対してグローバル・バッファを有効にした場合、実際のグローバル・セクションのサイズは 7968 ページであることがわかります。

必要となる GBLPAGES エントリの値を簡単に計算できる式はありません。この項の例からもわかるように、グローバル・バッファを有効化したときに割り当てられるグローバル・ページ・プールのサイズは、グローバル・バッファの数、ユーザー 1 人に割り当てられるグローバル・バッファの最大数、ユーザー数、データベース内にある記憶領域の数、After-image ジャーナルの有効化など、様々な要因によって変動します。このように様々な要因があるので、簡単な式を使って計算することは困難です。グローバル・セクションに必要なグローバル・ページの計算では、「Buffer Information」画面のグローバル・セクション・サイズを使用するのが最も良い方法です（グローバル・セクションに十分なページを確保するために、10 ～ 15 ページを必ず加算してください。

ユーザー数は、データベース管理者（DBA）が管理するデータベース・パラメータです。システムのリソースやパフォーマンスに実質的な影響を与えない場合、DBA は、一部のデータベース・パラメータをデフォルト値のままにしておきます。ユーザー総数のパラメータを明示的に指定しないでデータベースを作成すると、デフォルトでは 160 に設定されます。ほとんどのアプリケーションでは、特にシングル・ノードで実行しているアプリケーションやトランザクション処理モニターを実行しているアプリケーションでは、このように大きな値が要求されることはありません。

メモリーが十分にない場合は、ユーザー総数をシステム環境にあった値に手動で調整し、メモリーを確保してください。

次の表から、ユーザー数の増加に伴って、割り当てられるグローバル・ページの数も増えることがわかります。

```
Request made for 500 buffers at 6 blocks each
Number of users      Global pages allocated.
       20                    3488
```

60	3580
80	3626
100	3674
200	3904
300	4136
400	4366
500	4598

グローバル・バッファを無効にした状態で、グローバル・バッファやデータベース・パラメータを設定し、Performance Monitor の「Buffer Information」画面に表示されるグローバル・セクションの見積り値から、データベースのグローバル・セクションで必要になるグローバル・ページの数を予測します。また、「Buffer Information」画面で表示されるグローバル・セクションの値は、データベース・ルート・ファイルに格納されている情報を元に計算した見積り値です。データベースのルート・ファイルに格納されているものとは別のグローバル・バッファ・パラメータを GLOBAL 修飾子で指定して RMU Open コマンドを実行すると、そのデータベースのグローバル・セクションのサイズが増減します。

Oracle Rdb は、データベースをオープンした各ノード上で、データベースのグローバル・セクションをマッピングします。したがって、ノードからデータベースをオープンする場合、各ノードでの GBLSECTIONS、GBLPAGES、GBLPAGFIL、VIRTUALPAGECNT および PGFLQUOTA パラメータの要件を確認する必要があります。

GBLPAGFIL パラメータ

SYSGEN GBLPAGFIL パラメータは、ページ・ファイルのバックアップ格納域を使用した場合に、システム上に作成できるグローバル・ページの最大ページ数を定義します。GBLPAGFIL の値は、データベースの数、実行ユニットの数、グローバル・バッファの数とサイズ、オーバーヘッドなど、様々な要因をもとに決定します。

データベースに必要な GBLPAGFIL エントリ数を決定するには、Performance Monitor の「Buffer Information」画面で、データベースのグローバル・セクションに必要なグローバル・ページ数を検索してください。データベースで必要になる GBLPAGFIL エントリ数は、データベースで必要になるグローバル・ページ数と同じです。

同時に複数のデータベースを使用する場合、各データベースで必要になる値を計算してください。GBLPAGFIL パラメータの変更内容を有効にするためには、システムをリブートする必要があります。

GBLPAGFIL パラメータの値が小さすぎると、Oracle Rdb では、グローバル・バッファを割り当てた状態でデータベースをオープンできません。次のようなエラーが表示されます。

```
%RDMS-F-CANTOPENDB, database could not be opened as requested
-RDMS-F-CANTCREGBL, error creating and mapping database global section
-SYSTEM-F-EXGBLPAGFIL, exceeded global page file limit
%RMU-W-FATALERR, fatal error on <db-name>
```

Oracle Rdb が作成したグローバル・セクションがグローバル・ページ・ファイル・セクションを構成することから、このセクションが使用するグローバル・ページ数は、GBLPAGFIL

システム値から差し引かれます。したがって、GBLPAGFIL には、データベースのグローバル・セクションで必要なグローバル・ページ数に十分対応できる値を設定してください。

System Dump Analyzer（SDA）ユーティリティでは、使用可能な GBLPAGFIL エントリ数を確認できます。SDA を使用するには、OpenVMS の CMKRNL 権限が必要です。CMKRNL 権限は強力なので、権限を必要としないユーザーには与えないでください。次の例は、SDA を使用して、使用可能な GBLPAGFIL エントリ数を確認する方法を示しています。

```
$ ANALYZE/SYSTEM
SDA> !
SDA> ! Examine the original SYSGEN setting
SDA> !
SDA> EVALUATE @SGN$GL_GBLPAGFIL
Hex = 00002FA8   Decimal = 12200     UCB$M_TEMPLATE+00FA8
SDA> !
SDA> ! Examine the available GBLPAGFIL
SDA> !
SDA> EVALUATE @MMG$GL_GBLPAGFIL
Hex = 000015A1   Decimal = 5537     UCB$M_UNLOAD+005A1
```

この例では、5537 を超えるグローバル・ページを必要とするデータベースはオープンできないので、前に説明したエラーが表示されます。

RMU Open コマンドでは、グローバル・バッファ数を減らすことができます。

```
$ RMU/OPEN/GLOBAL=(TOTAL=10,USER_LIMIT=5) <db-name>
```

SQL ALTER DATABASE 文で、グローバル・バッファ・プールのグローバル・バッファに、小さな値を指定してください。指定した値は、データベース・ルート・ファイルに格納されます。

```
SQL> ALTER DATABASE FILENAME mf_personnel
  1>    GLOBAL BUFFERS ARE ENABLED
  2>    (NUMBER IS 10);
```

ALTER DATABASE 文でも、グローバル・バッファを無効にできます。

```
SQL> ALTER DATABASE FILENAME mf_personnel
  1>    GLOBAL BUFFERS ARE DISABLED;
```

GBLPAGFIL エントリの値を増やすには、ページ・ファイルのサイズを大きくするか、ページ・ファイルを追加する必要があります。この操作は、OpenVMS System Generation（SYSGEN）ユーティリティで変更できます。引き続き、MODPARAMS.DAT ファイルや AUTOGEN コマンド・プロシージャを使用して、システム・パラメータを変更してください。

VIRTUALPAGECNT パラメータ

VIRTUALPAGECNT パラメータは、プロセスがマッピングできる仮想ページの総ページ数です。プロセスはグローバル・セクションのマッピングを行うため、グローバル・セクションの増大に伴って、プロセスによるマッピング可能な仮想ページの数も増大します。した

がって、GBLPAGES パラメータの値を増やす場合、VIRTUALPAGECNT パラメータも、同等の値だけ増やしてください。

VIRTUALPAGECNT パラメータの変更内容を有効にするには、システムをリブートする必要があります。

Oracle Rdb ユーザーは、データベース上でグローバル・バッファを有効にしていると、-LIB-F-INSVIRMEM、仮想メモリー不足エラーが発生することがあります。このエラー・メッセージは、プロセスの VIRTUALPAGECNT または PGFLQUOTA が小さすぎることを示します。

グローバル・バッファを有効にすると、モニター・プロセスが消費する仮想メモリーは、グローバル・バッファの数に比例します。

さらに、定義するグローバル・バッファの数が増えると、ユーザー 1 人に必要な仮想メモリーの容量も増大します。ユーザーは、仮想メモリーを使用して、バッファ・プールのマッピングを行います。

注意：この項では、Oracle Rdb Monitor が消費する仮想メモリーについて説明していますが、各ユーザー・プロセスには、仮想メモリーおよびページ・ファイル・クォータの上限が適用されるので注意してください。モニターは、クォータの問題を検出しにくい場所なので、この項ではモニターについて説明します。

モニターは、グローバル・バッファを有効にしたデータベースをクローズしたとき、仮想メモリーをすべて完全に解放しないので、仮想メモリー・エラーが発生する回数が多くなります。

次の手順に従って、仮想メモリー不足の問題を診断してください。

1. Oracle Rdb Monitor のプロセス識別子（Process Identifier：PID）を取得します。
   仮想メモリー不足エラーが発生している場合、最初に、RDMS_MONITOR プロセスの PID を取得してください。

```
$ SHOW SYSTEM
VAX/VMS V5.5-2  on node GNPIKE  28-MAY-1996 11:17:18.88   Uptime  8 04:32:17
  Pid    Process Name    State  Pri      I/O       CPU       Page flts Ph.Mem
20800201 SWAPPER         HIB     16        0   0 00:09:20.18         0      0
20800206 CONFIGURE       HIB     10      204   0 00:00:03.39       123    320
20800219 RDMS_MONITOR70  LEF     15      143   0 00:00:05.40     18238    195
```

2. 次に、モニターのアカウンティング情報を取得します。
   SHOW PROCESS/ACCOUNTING コマンドを実行すると、プロセスの接続時間、ユーザー情報、仮想メモリーの最大サイズを参照できます。

```
$ SHOW PROCESS /ACCOUNTING /ID=20800219
28-MAY-1996 11:21:14.87   User: SYSTEM           Process ID:   20800219
                          Node: GNPIKE           Process name: "RDMS_MONITOR70"
```

```
Accounting information:
 Buffered I/O count:        44  Peak working set size:       1501
 Direct I/O count:          99  Peak virtual size:         139072
 Page faults:            18261  Mounted volumes:                0
 Images activated:           0
 Elapsed CPU time:           0 00:00:05.41
 Connect time:               8 04:18:51.37
```

3. この内容に基づいて、問題を診断してください。
次のいずれかが問題になっている可能性があります。

- 仮想ページ・サイズが小さすぎる
  Peak virtual size が SYSGEN の VIRTUALPAGECNT の上限またはそれに近い値である場合、AUTOGEN で VIRTUALPAGECNT の値を増やし、システムをリブートしてください。

  ```
  $ MCR SYSGEN
  SYSGEN>  SHO VIRTUAL
  Parameter Name             Current    Default     Min.     Max.     Unit
  Dynamic
  --------------             -------    -------    -------  -------   ----  ---
  ----
  VIRTUALPAGECNT              139072       9216       512   1200000 Pages
  ```

  VIRTUALPAGECNT は、1 つのプロセスにマッピングできる最大ページ数です。SHOW PROCESS/ACCOUNTING コマンドが返す「Peak virtual size」の値は、プロセスに現在マッピングされているページ数を示します。
  この例では、SYSGEN で設定されている VIRTUALPAGECNT の Current の値は 139072 です。Peak virtual size は 139072 です。モニターは、VIRTUALPAGECNT で設定された仮想メモリーの上限に達しています。
  SYSGEN の Current 列は、システムが使用している値を示しています。Default、Min.、Max. は、パラメータの有効値の範囲を示します。SYSGEN の値は、ワーキング・セットのように、動的な値ではありません。Current の値は、手動で変更しない限り、固定された値です。動的な SYSGEN パラメータは、即時有効になります。動的でないパラメータは、次回のシステム・リブートで有効になります。

- モニターのページ・ファイルが小さすぎる
  仮想メモリー不足が発生する最も大きな原因は、モニターを起動するアカウントの PGFLQUOTA の値が小さすぎることです。デフォルトでは、モニターは SYSTEM アカウントから起動します。通常、グローバル・バッファの値が数千以上であるのに対して、SYSTEM アカウントの PGFLQUOTA には、かなり小さい値が設定されています。PGFLQUOTA は、プロセスの仮想メモリー管理のために、プロセスがシステム・ページング・ファイル内で使用できる最大ページ数です。PGFLQUOTA が不足すると、マッピングしたページを格納する場所がなくなるので、エラーが発生します。モニタープロセスでクォータの問題が発生したことが原因でデータベース接続が許可されなかった場合、モニターのログ・ファイルはエラーを報告します。

SYSGEN VIRTUALPAGECNT が、SHOW PROCESS/ACCOUNTING コマンドで表示した Peak virtual size を大幅に上回っている場合、上限に達している可能性があります。次の例は、モニター・アカウントの PGFLQUOTA パラメータの値を大きくする方法を示します。

```
$ SET DEF SYS$SYSTEM
$ MCR AUTHORIZE
UAF> MODIFY SYSTEM/PGFLQUO=400000
%UAF-I-MDFYMSG, user record(s) updated
UAF>
```

モニター・プロセスに十分なクォータが確保されていることを確認するには、RDM$MON_USERNAME 論理名を使用する方法もあります。論理名 RDM$MON_USERNAME は、モニター・プロセスが起動時に継承するクォータを所有しているユーザー名を指定します。RDM$MON_USERNAME 論理名の詳細は、A.78 項（A-34）を参照してください。
ページ・ファイルのサイズは、新しく設定したサイズに対応できる大きさを確保してください。次に、モニターを起動します（システム・アカウントから起動するか、適切なクォータが設定されたアカウントに RDM$MON_USERNAME 論理名を定義した後）。これで、新しいページ・ファイル・クォータが使用されます。
ユーザー・プロセスが仮想メモリー・クォータを超えるか、ページ・ファイル・クォータが不足している場合にも、問題が発生します。
仮想メモリーの問題を解決することによって、他のリソース不足を引き起こす場合もあります。

```
$ RMU/OPEN SQL$DATABASE/GLOBAL=(TOTAL=25000,USER=10)
%RDMS-F-CANTOPENDB, database could not be opened as requested
-RDMS-F-CANTCREGBL, error creating and mapping database global section
-SYSTEM-F-EXGBLPAGFIL, exceeded global page file limit
```

このような場合、SYSGEN の GBLPAGFIL パラメータの値を大きくする必要がある、という内容のエラー・メッセージが表示されます。GBLPAGFIL の値の調整については、この項の前半の GBLPAGFIL パラメータの説明を参照してください。

PGFLQUOTA パラメータ

PGFLQUOTA パラメータは、プロセスの仮想メモリー管理のために、プロセスがシステム・ページング・ファイル内で使用できる最大ページ数です。PGFLQUOTA が不足すると、マッピングしたページを格納する場所がなくなるので、仮想メモリー不足のエラーが発生します。仮想メモリー不足が発生する最も大きな原因は、モニターを起動するアカウントの PGFLQUOTA の値が小さすぎることです。デフォルトでは、モニターは SYSTEM アカウントから起動しますが、グローバル・バッファの値が数千以上であるのに対して、SYSTEM アカウントの PGFLQUOTA にはかなり小さい値が設定されています。

PGFLQUOTA パラメータの詳細と仮想メモリー不足エラーの分析については、この項の VIRTUALPAGECNT パラメータの説明を参照してください。◆

4.1.2.13 グローバル・バッファのパフォーマンス分析

一般的に、データベースが消費する CPU リソースは、グローバル・バッファを有効にすると、ローカル・バッファを有効にした場合よりおよそ 10% 増加します。CPU リソースに余裕がない環境では、グローバル・バッファを有効にする前に、CPU の使用率の増加分を考慮してください。

グローバル・バッファを有効にすると、ローカル・バッファを有効にした場合よりもページ・フォルトの数が多くなります。GLOBAL VALID ページ・フォルトの値が大きくなることは、フォルトの原因になっているページがグローバル・バッファ・プール内に存在することを示しています。これはソフト・フォルトであり、問題はありません。

OpenVMS VAX OpenVMS Alpha

ただし、GLOBAL VALID ページ・フォルトの数値を小さくしたい場合は、データベースを使用しているプロセスのワーキング・セット・クォータ値を大きくします。詳細は、8.2 項 (8-38) を参照してください。OpenVMS の MONITOR PAGE コマンドは、カテゴリ別にページ・フォルトを区分しています。

グローバル・バッファでは、ローカル・バッファよりもロック操作が多く発生します。ロック操作には、システム所有のロックであり、リソースを多く消費するものと、ローカルのロックがあります。システム所有のページ・ロックは、VMScluter 内のノード間で、ページ・バージョン番号を保持します。グローバル・バッファの各ページには、ロックが対応付けられています。このシステム所有のロックは、ENQLM を消費することはありませんが、ユーザーの割当てセットに含まれる各ページには、ENQLM に対するロックが存在します。システム所有のロックは、LOCKIDTBL エントリを使用します。したがって、グローバル・バッファのページ数が多い場合、LOCKIDTBL（LOCKIDTBL_MAX,、SRPCOUNT および SPRCOUNTV も同様）の値に影響します。リソースをあまり消費しないローカル・ロックは、グローバル・バッファのデータ構造へのアクセスを同期化します。ロックの増加は、ほとんどがページ・ロックであり、レコード・ロックの場合もあります。データベース管理者は、グローバル・バッファに関連するロックが増加しても対応できるように、LOCKIDTBL、LOCKIDTBL_MAX、SRPCOUNT および SPRCOUNTV に十分なエントリを確保してください。OpenVMS の MONITOR LOCK コマンドを実行すると、システム上のロックの総数が表示されるので、この値と LOCKIDTBL エントリを比較してください。◆

グローバル・バッファとローカル・バッファのパフォーマンスをテストしたい場合は、同等化を念頭に置いてください。同等化とは、ローカル・バッファとグローバル・バッファの設定が同等であるときの設定内容を指します。同等化の基準となるのはメモリー使用量であり、次のような 2 つのルールを導くことができます。

- すべてのユーザーが使用するバッファの数の合計は、ローカル・バッファとグローバル・バッファで同じ値でなければなりません。
- 各ユーザーが使用するバッファ数は、ローカル・バッファとグローバル・バッファで同じ値でなければなりません。

1 つの方法として、バッファ・パラメータは次のように設定できます。

1. NUMBER OF BUFFERS パラメータ（バッファのデフォルト数）を B に設定します。

2. USER LIMIT パラメータ（ユーザー 1 人あたりのグローバル・バッファの最大数）も B に設定します。
3. パフォーマンス・テストを実施するユーザー数が N だとすると、NUMBER IS パラメータ（グローバル・バッファの合計数）には、B × N を設定します。

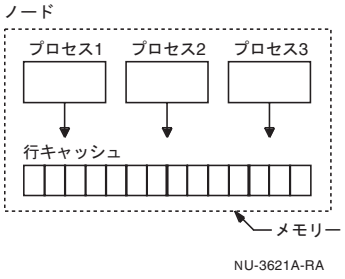
簡略化するために、バッファ論理名や構成パラメータは使用しないでください。このようなガイドラインに従って、グローバル・バッファの有効化と無効化や、パフォーマンス・テストを実施してください。

4.1.3 行キャッシュ

行キャッシュ は、グローバル・アクセスが可能なメモリー領域であり、行のコピーが格納されています。行キャッシュにより、頻繁にアクセスが発生する行をメモリーに格納し、ディスク I/O を軽減できます。対応付けられたページがディスクにフラッシュされた後も、行はメモリー内に常駐します。行キャッシュには、次のような利点があります。

- データベース・ページの読取りと書込みを軽減
- 応答時間を向上
- グローバルまたはローカル・バッファ内のページと比べて、行キャッシュ内の行へのアクセスではオーバーヘッドが大幅に軽減
- 行が行キャッシュ内で見つかると、コード・パスが短縮
- データ共有により、システム・リソース（メモリー）を効率的に使用

行キャッシュには、インデックス構造やテーブル・データを格納できます。行キャッシュは、次のように、データベースに接続しているすべてのプロセスが共有します。



ユーザーが行を要求すると、Oracle Rdb は、最初に、要求された行が既存の行キャッシュにマッピングされていないかどうかを調べます。行キャッシュがマッピングされている場合、Oracle Rdb は、要求された行キャッシュに存在するかどうかを調べます。行が行キャッシュに存在すれば、その行を取得します。行が行キャッシュに存在しない場合、Oracle Rdb は、ページ・バッファ・プールを調べます。行がページ・バッファ・プールに存在しない場合、

Oracle Rdb は、ディスク I/O を実行して行を取得します。要求された行は、空き領域があれば、行キャッシュに挿入されます。ページ・バッファ・プールやディスクからの取出しと比べて、行キャッシュから行を取り出す方が、コード・パスは短縮されます。

たとえば、次のようなキャッシングによって、パフォーマンスを向上できます。

- 複数のユーザーが頻繁にアクセスする共有データ
- B ツリー・インデックスのリーフ以外のノード
  リーフ以外のノードのキャッシングにより、メモリー内のインデックスを効率的に格納できます。
- RDB$SYSTEM 記憶領域
  RDB$SYSTEM 記憶領域のすべてをキャッシングすることにより、メタデータ・クエリーのパフォーマンスが向上します。

データベースの記憶領域を効率的に分割しておけば、どのデータをキャッシングすればよいかを判断するタスクも簡単になります。データの内容がわかっていれば、どのテーブル行やインデックスをキャッシングすればよいかを効率的に判断できます。

行キャッシュには、次の 2 つのタイプがあります。

- 物理領域
  物理領域キャッシュは、記憶領域に定義されています。物理領域キャッシュには、1 つ以上の記憶領域のデータを格納できます。物理領域キャッシュを記憶領域に明示的に割り当てるには、CREATE STORAGE AREA 文の CACHE USING 句、または CREATE DATABASE 文や ALTER DATABASE 文の ADD STORAGE AREA 句を使用します。
  物理領域キャッシュは、システム・レコードをキャッシュします。さらに、物理領域キャッシュを定義すれば、指定した記憶領域内にある様々なサイズの行すべてが、行キャッシュの候補になります。
- 論理領域
  論理領域キャッシュは、テーブルやインデックスに定義されています。Oracle Rdb は、名前を基準に、テーブルやインデックスに論理領域キャッシュを自動的に割り当てます。論理領域キャッシュの名前は、キャッシングしたいテーブルやインデックスの名前と同じでなければなりません。
  1 つのキャッシュに、同等サイズの行を格納することにより、論理領域キャッシュによってかなり大量の領域を節約できます。

4.1.3.1 行キャッシュの要件

行キャッシュを使用するためには、次のような条件を満たす必要があります。

- クラスタ・ノードの数は 1 つ
- After-image ジャーナルを有効化
- 高速コミットを有効化

- 1 つ以上のキャッシュ・スロットを予約
- 行キャッシングを有効化

RMU Dump Header コマンドを実行し、行キャッシュを使用する上で必要となる条件を満たしているか確認してください。次のコマンド出力は、満たされていない各項目に対して、警告を表示します。

```
.
.
.
 Row Caches...
   - Active row cache count is 0
   - Reserved row cache count is 1
   - Sweep interval is 1 second
   - Default cache file directory is ""
   - WARNING: Maximum node count is 16 instead of 1
   - WARNING: After-image journaling is disabled
   - WARNING: Fast commit is disabled
.
.
.
```

4.1.3.2 行キャッシュの有効化

行キャッシュを有効にするには、ROW CACHE IS ENABLED 句を指定して、SQL ALTER DATABASE 文と CREATE DATABASE 文を実行します。

```
SQL> ALTER DATABASE FILENAME mf_personnel
  1> ROW CACHE IS ENABLED;
```

行キャッシュを無効にするには、ROW CACHE IS DISABLED 句を指定して、SQL ALTER DATABASE 文と CREATE DATABASE 文を実行します。

```
SQL> ALTER DATABASE FILENAME mf_personnel
  1> ROW CACHE IS DISABLED;
```

4.1.3.1 項（4-62）で説明した条件が 1 つでも満たされない場合、行キャッシュは無効化されます。

行キャッシュを無効にすると、すでに作成し、割当て済の行キャッシュはそのまま保持され、次に行キャッシュが有効化されたときに使用されます。

行キャッシュの有効化と無効化は、データベースにユーザーが接続していない状態で実行してください。

行キャッシュが有効化されていることを確認するには、Performance Monitor を起動し、メイン・メニューを見てください。次は、行キャッシュが有効化されている状態でのメイン・メニューです。

```
                              Select Display
   A. Summary IO Statistics              O. IO Statistics (by file)  [->
   B. Summary Locking Statistics         P. Locking (one lock type)  [->
   C. Summary Object Statistics          Q. Locking (one stat field) [->
   D. Summary Cache Statistics           R. Lock Statistics (by file)[->
   E. Summary Cache Unmark Statistics    S. Database Parameter Info  [->
   F. Record Statistics                  T. Row Cache (One Cache)    [->
   G. Transaction Duration (Total)       U. Row Cache (One Field)    [->
   H. Custom Statistics                  V. Row Cache Information    [->
   I. Snapshot Statistics                W. Index Information        [->
   J. Process Information       [->      X. General Information      [->
   K. Journaling Information    [->      Y. Objects (one stat type)  [->
   L. Hot Standby Information   [->      Z. Objects (one stat field) [->
   M. IO Statistics             [->      0. Database Dashboard       [->
   N. Global Buffer Information[->       1. Online Analysis & Info.  [->
```

4.1.3.1 項（4-62）で説明した条件が 1 つでも満たされていないと、メイン・メニューには、行キャッシュの画面は表示されません。

4.1.4 行キャッシュ情報の収集

例 4-11（4-64）では、RMU Dump Header コマンドを使用して、行キャッシュに関する情報を表示しています。

例 4-11 行キャッシュ・パラメータ

```
$ RMU/DUMP/HEADER mf_personnel
   .
   .
   .
    Row Caches... (1)
      - Active row cache count is 23
      - Reserved row cache count is 54
      - Sweep interval is 1 second
      - Default cache file directory is ""
   .
   .
   .
Storage area "EMPIDS_LOW"
   .
   .
   .
    Row Caching... (2)
      - Row caching is enabled
      - Row cache ID is 1
   .
   .
```

```
   .
Row cache "EMPIDS_LOW"
    Cache ID number is 1 (3)
    Allocation... (4)
      - Row slot count is 1000
      - Maximum row size allowed in cache is 200 bytes
      - Working set count is 10
      - Maximum slot reservation count is 20
      - Row replacement is enabled
    Files... (5)
      - No file directory has been specified
      - File allocation is 100 blocks
      - File extension is 100 blocks
    Shared Memory... (6)
      - System space memory is disabled
      - Large memory is disabled
      - Large memory window count is 100
    Hashing... (7)
      - Hash value for logical area DBIDs is 211
      - Hash value for page numbers is 11
    Checkpointing...  (8) (4-68)
      - Last checkpoint is 0:2
      - Checkpoint sequence is 2
   .
   .
   .
```

次の番号は、例 4-11（4-64）で示したパラメータの説明です。

(1) 概要情報

- Active row cache count is 23
  この値は、このデータベースで現在定義されている行キャッシュの数を示します。
- Reserved row cache count is 54
  この値は、データベースで使用可能なスロットの数を示します。キャッシュ・スロットは、ALTER 文または CREATE DATABASE 文の RESERVE n CACHE SLOTS パラメータで予約します。
- Sweep interval is 1 second
  スイープ間隔は、ROW CACHE 文の SWEEP INTERVAL パラメータで指定します。スイープとは、アクティブな行キャッシュすべてを検査して、変更された行を記憶領域に書き戻す処理です。
- Default cache file directory is ""
  デフォルトのキャッシュ・ファイル・ディレクトリは、ディレクトリを明示的に指定しない場合に、Oracle Rdb がキャッシュ・バックアップ格納域ファイルを格納するディレクトリです。ディレクトリの指定は、次の SQL 構文を使用します。

- CREATE または ADD CACHE 句の LOCATION パラメータ
- CREATE または ALTER DATABASE 文の ROW CACHE 句の LOCATION パラメータ

格納先を指定しないと、Oracle Rdb は、データベース・ルート・ファイルのディレクトリにキャッシュ・バックアップ格納域ファイルを格納します。
バックアップ格納域ファイルには、OpenVMS では行キャッシュのグローバル・セクションの内容、Compaq Tru64 UNIX では共有メモリー・パーティションの内容に関する情報が格納されています。

(2) 記憶領域情報

- Row caching is enabled
  これは、ROW CACHE パラメータで指定されています。行キャッシュを有効にするには、行キャッシュを定義し、1 つ以上の記憶領域に割り当ててください。デフォルトでは、行キャッシュは無効です。
- Row cache ID is 1
  Oracle Rdb は、データベース内で定義した各行キャッシュに ID を割り当てます。

(3) Cache ID number is 1
Oracle Rdb は、データベース内で定義した各行キャッシュに ID を割り当てます。

(4) Allocation...

- Row slot count is 1000
  これは、CACHE SIZE is n ROWS パラメータで指定されます。
- Maximum row size allowed in cache is 200 bytes
  これは、ROW LENGTH is n BYTES パラメータで指定されます。
- Working set count is 10
  これは、置き換えられない使用中の行の数です。
- Maximum slot reservation count is 20
  これは、NUMBER OF RESERVED ROWS パラメータで指定されます。デフォルト値は、20 行です。
  予約済の行数は、Oracle Rdb が各プロセスに予約するキャッシュ内にスロットがいくつ存在するかを示しています。行を数多く予約することにより、行をキャッシュに挿入する間に発生する行キャッシュのロックを最小限に抑えることができます。
- Row replacement is enabled
  これは、ROW REPLACEMENT パラメータで指定されます。デフォルトでは、行の置換は有効です。

(5) Files...

- No file directory has been specified
  LOCATION パラメータは、キャッシュ・バックアップ格納域ファイルのディレクトリを指定します。行キャッシュ・サーバー（Row Cache Server：RCS）がチェックポイントを処理するときに、Oracle Rdb は、キャッシュ・バックアップ格納域ファイルを書き込みます。Oracle Rdb は、拡張子 .rdc でファイルを自動的に作成します。キャッシュ・バックアップ格納域ファイルのデフォルトの格納先は、データベース・ルート・ファイルが格納されているディレクトリです。
  LOCATION パラメータは、データベース・レベルまたは行キャッシュ・レベルで指定できます。LOCATION パラメータを ROW CACHE 句で指定すると、指定したディレクトリが、データベースで定義したすべての行キャッシュのデフォルト・ディレクトリになります。ただし、行キャッシュの定義で、LOCATION パラメータを指定すれば、個々の行キャッシュで使用するデフォルト・ディレクトリを上書きできます。
- File allocation is 100 blocks
  ALLOCATION パラメータは、.rdc ファイルの初期サイズを指定します。デフォルトの割当ては、キャッシュ・サイズの 40% です。キャッシュ・サイズは、キャッシュ内の行数に行の長さを掛けた値です。
- File extension is 100 blocks
  EXTENT パラメータは、キャッシュ・バックアップ格納域ファイル（.rdc）が最初の割当ての上限に達した後、拡張可能なページ数を指定します。デフォルトは、キャッシュ内の行数に 127 を掛けた値です。

(6) Shared Memory...

OpenVMS Alpha

- System space memory is disabled
  これは、SHARED MEMORY パラメータで指定されます。Oracle Rdb が、共有メモリー内に行キャッシュを作成するかどうかを指定します。デフォルトでは、行キャッシュはプロセスのグローバル・セクション内に作成されます。◆

OpenVMS Alpha

- Large memory is disabled
  これは、LARGE MEMORY パラメータで指定されます。Oracle Rdb が、物理メモリー内に行キャッシュを作成するかどうかを指定します。デフォルトでは無効です。◆
- Large memory window count is 100
  これは、WINDOW COUNT パラメータで指定されています。デフォルト値は、100 ウィンドウです。WINDOW COUNT は、仮想アドレス空間にある各ユーザーのプライベート・ウィンドウにマッピングする物理メモリー・ロケーションの数を指定します。

(7) Hashing...

- Hash value for logical area DBIDs is 211
- Hash value for page numbers is 11

ハッシュ値は、行キャッシュのハッシュ・テーブル・キューの分散を、Oracle Rdb が精密にチューニングするために使用します。

(8) Checkpointing...

- Last checkpoint is 0:2
  最後のチェックポイントの AIJ 順序番号と AIJ 仮想ブロック番号（Virtual Block Number：VBN）を指定します。
- Checkpoint sequence is 2

4.1.4.1 行キャッシュの作成と使用方法

行キャッシュ機能を使用するためには、次の手順を実行してください（または、Oracle Rdb のデフォルト値をそのまま使用してください）。

1. 行キャッシュのポインタ用に、データベース・ルート・ファイルのスロットを予約します。
   行キャッシュにスロットを予約すると、データベースがオンラインの状態でも行キャッシュを追加できます。スロットを予約しても、行キャッシュは有効化されないので注意してください。行キャッシュ・スロットの予約については、4.1.4.1.1 項（4-68）を参照してください。
2. 行キャッシュを作成します。
   SQL CREATE DATABASE STATEMENT 文の CREATE CACHE 句、または SQL ALTER DATABASE 文または IMPORT 文の ADD CACHE 句を使用して、行キャッシュを作成します。キャッシュに格納する行の数とサイズを指定することもできます。行キャッシュの作成の詳細は、4.1.4.1.2 項（4-69）を参照してください。
3. メモリーを選択します。
   キャッシュの格納先となるメモリー領域を指定します。詳細は、4.1.4.1.3 項（4-71）を参照してください。
4. 行キャッシュを記憶領域に割り当てます。
   行キャッシュを記憶領域に割り当てるには、CREATE STORAGE AREA 文の CACHE USING 句、または、CREATE DATABASE 文または ALTER DATABASE 文の ADD STORAGE AREA 句を使用します。必要な操作は、物理領域キャッシュの割当てのみです。Oracle Rdb は、論理領域キャッシュを自動的に割り当てます。詳細は、4.1.4.1.4 項（4-75）と 4.1.4.1.5 項（4-75）を参照してください。

4.1.4.1.1 行キャッシュのスロットの予約 データベースの作成では、将来的に必要になる行キャッシュの数や、それだけの行キャッシュに十分対応できる数のスロットを予約することを考慮してください。スロットの予約では、行キャッシュへのポインタ用に、データベー

ス・ルート・ファイルにスロットを予約します。行キャッシュに十分な数のスロットを予約しておけば、データベースがオンラインの状態でも行キャッシュを追加できるので、データベース処理を中断する必要がありません。

行キャッシュでのスロット予約は、次の例のように、ALTER DATABASE 文の RESERVE n CACHE SLOTS 句を使用します。

```
SQL> ALTER DATABASE
  1>  FILENAME 'mf_personnel'
  2>  RESERVE 20 CACHE SLOTS;
```

RESERVE n CACHE SLOTS 句を指定しないと、Oracle Rdb は、スロットを 1 つ予約します。

4.1.4.1.2 行キャッシュのサイズ指定 行キャッシュの作成や行キャッシュ定義の変更では、次のような内容をオプションで指定できます。

- スロット・サイズ
  スロット・サイズ は、行キャッシュに格納できる最長行のサイズです。行サイズが大きすぎてキャッシュに格納できない場合、Oracle Rdb は、行をキャッシュしません。
  キャッシュ内の最長行のサイズを指定するには、ADD、ALTER、CREATE CACHE 句の ROW LENGTH IS パラメータを使用します。
- スロット・カウント
  スロット・カウント は、キャッシュ内に格納できる行の数です。キャッシュに格納可能な行数を指定するには、ADD、ALTER、CREATE CACHE 句の CACHE SIZE IS パラメータを使用します。

次は、行キャッシュの定義の例を示しています。

```
SQL> ADD CACHE RCACHE_1
  1> ROW LENGTH IS 200 BYTES
  2> CACHE SIZE IS 3000 ROWS;
SQL> --
SQL> -- In this example, the slot size is 200
SQL> -- and the slot count is 3000.
SQL> --
```

ROW LENGTH 句を指定しないと、Oracle Rdb は、行が最長 256 バイトのキャッシュを作成し、CACHE SIZE 句を指定しないと、最大 1000 行のキャッシュを作成します。

Oracle Rdb は、行の長さに指定された値が 4 で割り切れない場合、4 バイトごとの境界に切り上げます。これは、ロングワードを単位にしたデータ構造が最適なパフォーマンスを発揮できるからです。

行キャッシュに適切なサイズを選択することは、非常に重要です。これまでに説明したとおり、行サイズが大きすぎると、Oracle Rdb は行をキャッシュしません。Oracle Rdb は、ディスクから行を取り出す前に、キャッシュ内に行が存在するかどうかを確認するので、行サイズが大きくなるほどシステム・パフォーマンスは低下します。RMU Dump Areas コマ

ンドを実行すると、データ行、ハッシュ・バケット、B ツリー・ノードのサイズを確認できます。テーブル内の行サイズは大きく変動する点に注意してください。たとえば、テーブルに格納されている行の最大長が 100 バイトであっても、大部分の行が 40 ～ 50 バイトであれば、スロット・サイズに 100 バイトを選択する必要はありません。ただし、オーバーヘッドを含め、ほとんどの行を考慮に入れてください。テーブル内にある他の行サイズと比較しないで最大値をそのまま選択してしまうと、大量にメモリーを浪費する可能性があります。

次の例は、MY_AREA 記憶領域から数ページをダンプする例です。

```
$ RMU/DUMP/AREA=MY_AREA/START=5/END=10 test_db/OUT=rmu_dump_area.out
```

rmu_dump_area.out ファイルで、"total hash bucket" と "static data" を検索してください。

```
$ SEARCH rmu_dump_area.out "total hash bucket"
                                ....  total hash bucket size: 97
                                ....  total hash bucket size: 118
                                ....  total hash bucket size: 118
                                ....  total hash bucket size: 118
                                ....  total hash bucket size: 118
                                ....  total hash bucket size: 118
                                ....  total hash bucket size: 118
                                ....  total hash bucket size: 118
                                ....  total hash bucket size: 118
   .
   .
   .
$ SEARCH rmu_dump_area.out "static data"
                                ....  311 bytes of static data
                                ....  311 bytes of static data
                                ....  311 bytes of static data
                                ....  311 bytes of static data
                                ....  311 bytes of static data
                                ....  311 bytes of static data
                                ....  311 bytes of static data
                                ....  311 bytes of static data
                                ....  311 bytes of static data
                                ....  311 bytes of static data
                                ....  311 bytes of static data
                                ....  311 bytes of static data
   .
   .
   .
```

ハッシュ・バケット・サイズは 118 バイト、データ行サイズは 311 バイトです。テーブル内の他の行でも、この程度の領域が必要になります。適切な行サイズを決定するときに重要なのは、サンプル・ページをランダムにスキャンすることです。Oracle Rdb は、行サイズを、最も近いロングワードに切り上げます。

Performance Monitor の行キャッシュに関する画面では、キャッシュへの行の挿入に関する統計を表示します。統計の 1 つである "row too big" は、行が大きすぎて指定のキャッシュに格納できないことを示します。行キャッシュに関連する Performance Monitor の詳細は、4.1.4.6 項（4-86）を参照してください。

スロット・カウントにスロット・サイズを掛けると、行キャッシュ・サイズの概算値をバイト単位で計算できます。この数値には、オーバーヘッドも加算してください。

4.1.4.1.3 メモリーの割当て 行キャッシュの作成や行キャッシュ定義の変更では、メモリー内でキャッシュを作成する場所を指定できます。行キャッシュは、次のようなメモリー位置に格納できます。

- OpenVMS 上のプロセス・グローバル・セクションまたは Compaq Tru64 UNIX 上の共有メモリー・パーティション
  プロセス領域内に作成したグローバル・セクションを使用する場合、すべてのユーザーが物理メモリーを共有し、OpenVMS オペレーティング・システムは、各ユーザーのプライベート・アドレス空間にキャッシュをマッピングします。
  SHARED MEMORY IS PROCESS パラメータを使用して、キャッシュの作成先を、プロセス・グローバル・セクションと共有メモリー・パーティションのいずれかに指定してください。次に例を示します。

```
SQL>
SQL> ALTER DATABASE FILENAME mf_personnel
  1> ADD CACHE EMPIDS_LOW_RCACHE
  2> SHARED MEMORY IS PROCESS;
```

  これはデフォルトです。

OpenVMS Alpha

- システム領域バッファ
  システム領域のグローバル・セクションが OpenVMS Alpha システム領域に存在する場合、システム領域のグローバル・セクションがメモリー内に常駐、つまり固定されていることを意味し、プロセスのワーキング・セットのサイズには影響しません。
  システム領域は、システム全体に重要な影響を与えます。システム領域バッファは、ページングしないで物理メモリーを使用するので、他のシステム・タスクが使用できる物理メモリーの容量が減ってしまいます。これは、メモリー容量に制限があるシステムでは問題になります。システム領域は、慎重に割り当ててください。非ページング動的プール（Nonpaged dynamic pool：NPAGEDYN）と VAXcluster キャッシュ（VCC）は、システム領域を使用するシステム・パラメータの例です。
  SHARED MEMORY IS SYSTEM パラメータを使用して、キャッシュの作成先としてシステム領域バッファを指定します。次に例を示します。

```
SQL>
SQL> ALTER DATABASE FILENAME 'mf_personnel'
  1> ADD CACHE EMPIDS_MID_RCACHE
  2> SHARED MEMORY IS SYSTEM;
```

頻繁にアクセスされるデータを格納したサイズの小さなバッファを、システム領域バッファに割り当てることを検討してください。行キャッシュをシステム領域バッファに格納すると、データをユーザー・ウィンドウにマッピングする必要がないので、プロセスのオーバーヘッドがなく、データ・アクセスは非常に高速になります。Performance Monitor の「Hot Row Information」画面では、特定の行キャッシュについて、最も頻繁にアクセスされる行を一覧表示します。◆

OpenVMS Alpha

- Very large memory（VLM）
  Very large memory（VLM）では、システム上で使用可能なすべての物理メモリーを利用でき、データベース・ユーザーの仮想アドレス空間への動的マッピングが可能です。VLM では、小規模な仮想アドレス・ウィンドウを使用して、大量の物理メモリーにアクセスします。VLM は物理メモリー内で定義されていますが、仮想アドレス・ウィンドウは、ユーザーのプライベート仮想アドレス空間で定義および格納されています。VLM にキャッシュを作成するには、LARGE MEMORY パラメータを使用します。

```
SQL>
SQL> ALTER DATABASE FILENAME 'mf_personnel'
  1> ADD CACHE EMPIDS_OVER_RCACHE
  2> LARGE MEMORY IS ENABLED;
SQL>
```

◆

VLM は、サイズの大きなテーブルに頻繁にアクセスする場合に便利です。VLM を使用する上で制約となるのは、システム上で使用可能な物理メモリーの容量です。

物理メモリーは、ウィンドウを使用して表示できます。WINDOW COUNT パラメータで、ウィンドウ・ペインの数を指定できます。Oracle Rdb は、デフォルトで、1 つのプロセスに 100 のウィンドウ・ペインを割り当てます。

表 4-5（4-72）は、行キャッシュ・オブジェクトのメモリー位置と、プロセスのプライベート仮想アドレス・ウィンドウがデータ・アクセスに必要かどうかを示しています。

表 4-5 行キャッシュ・オブジェクトのメモリー位置

SHARED	LARGE	制御構造	データ行	ウィンドウ
PROCESS[*1]	DISABLED[*3]	プロセス・グローバル・セクションまたは共有メモリー・パーティション	プロセス・グローバル・セクションまたは共有メモリー・パーティション	なし
PROCESS[*1]	ENABLED[*4]	プロセス・グローバル・セクションまたは共有メモリー・パーティション	物理メモリー	あり
SYSTEM[*2]	DISABLED[*3]	システム領域	システム領域	なし

表 4-5 行キャッシュ・オブジェクトのメモリー位置（続き）

SHARED	LARGE	制御構造	データ行	ウィンドウ
SYSTEM[*2]	ENABLED[*4]	システム領域	物理メモリー	あり

[*1] SHARED MEMORY IS PROCESS

- 行キャッシュの制御構造は、プロセス・グローバル・セクションまたは共有メモリー・パーティション内にあります。
- データ行のストレージは、LARGE MEMORY を有効にするか無効にするかによって決まります。
  - LARGE MEMORY を有効にすると、データは物理メモリー内に格納され、データ・アクセスには、各ユーザーのプロセス仮想アドレス空間からウィンドウが必要になります。
  - LARGE MEMORY を無効にすると、データはプロセス・グローバル・セクションまたは共有メモリー・パーティションに格納され、データ・アクセスにはウィンドウは必要ありません。

[*2] SHARED MEMORY IS SYSTEM

- 行キャッシュの制御構造は、システム領域に格納されます。
- データ行のストレージは、LARGE MEMORY を有効にするか無効にするかによって決まります。
  - LARGE MEMORY を有効にすると、データは物理メモリー内に格納され、データ・アクセスには、各ユーザーのプロセス仮想アドレス空間からウィンドウが必要になります。
  - LARGE MEMORY を無効にすると、データはシステム領域に格納され、データ・アクセスにウィンドウは必要ありません。

[*3] LARGE MEMORY IS DISABLED

- データ行と行キャッシュ制御構造のストレージは、SHARED MEMORY が PROCESS か SYSTEM かによって決まります。
  - SHARED MEMORY IS PROCESS の場合、データと行キャッシュ制御構造は、プロセス・グローバル・セクションまたは共有メモリー・パーティションに格納され、データ・アクセスにウィンドウは必要ありません。
  - SHARED MEMORY IS SYSTEM の場合、データと行キャッシュ制御構造はシステム領域に格納され、データ・アクセスにウィンドウは必要ありません。

*4LARGE MEMORY IS ENABLED

- データ行は物理メモリー内に格納され、データ・アクセスには、プロセスのプライベート仮想アドレス・ウィンドウが必要になります。
- 行キャッシュ制御構造のストレージは、SHARED MEMORY が PROCESS か SYSTEM かによって決まります。
  - SHARED MEMORY IS PROCESS の場合、制御構造は、プロセスのグローバル・セクションまたは共有メモリー・パーティション内に格納されます。
  - SHARED MEMORY IS SYSTEM の場合、制御構造は、システム領域に格納されます。

行キャッシュの作成や使用を始める場合は、まず最初に、使用可能なメモリー容量を検討してください。ただし、現在のところ、システム上で使用可能なシステム領域の容量を測定できるツールはありません。VLM 行キャッシュは、仮想アドレス・ウィンドウ用にシステム領域を使用するので、オラクル社は、最初に VLM キャッシュを定義してアクティブにしておくことをお薦めします。これにより、VLM システム領域の要件を満たしてから、システム領域バッファの行キャッシュをアクティブにできます。VLM 行キャッシュを定義したら、頻繁にアクセスするデータを格納した小さなテーブル用に、システム領域バッファの行キャッシュを定義してください。

OpenVMS VAX OpenVMS Alpha

OpenVMS システムでは、DCL コマンド SHOW MEMORY/PHYSICAL を使用して、物理メモリーの使用可能なサイズや使用状況を確認できます。このコマンドを実行すると、使用済のメモリー容量と空き容量に関する情報が表示されます。空きメモリーは、ユーザー・アプリケーションだけでなく、VLM 行キャッシュでも使用できます。

次の例は、1.5GB のメモリーを搭載し、OpenVMS Alpha のメモリー・ページの合計が 196608 のシステムです（OpenVMS Alpha ページは 8192 バイト）。

```
$ SHOW MEMORY/PHYSICAL
             System Memory Resources on 29-MAY-1996 21:39:35.40
Physical Memory Usage (pages):     Total        Free      In Use    Modified
  Main Memory (1536.00Mb)         196608      183605       12657         346
```

1.5GB のうち、183605 ページは空きリスト上に残っています。空きメモリーのほとんどは、行キャッシュの割当てに使用できます。

MY_TABLE テーブルには、論理領域キャッシュが定義されていると仮定します。次の SQL 文は、論理領域キャッシュのマッピングを行います。

```
SQL> ATTACH 'FILE test_db';
SQL> SELECT * FROM MY_TABLE WHERE MY_HASH_INDEX = 100;
```

この SQL 文を発行すると、論理領域キャッシュは、40462 の OpenVMS Alpha ページに対して、必要なメモリー・アカウンティングを割り当てます。次の SHOW MEMORY/PHYSICAL コマンド出力を参照してください。

```
$ SHOW MEMORY/PHYSICAL
```

```
                System Memory Resources on 29-MAY-1996 21:46:07.01
Physical Memory Usage (pages):     Total        Free      In Use    Modified
  Main Memory (1536.00Mb)         196608      143143       52766         699
```

また、空きメモリーの容量が少なくなっていることにも注意してください。

ユーザーがデータベースの接続、ワーキング・セットの割当て、作業を開始した後、次のSHOW MEMORY/PHYSICAL コマンドを発行しました。

```
                System Memory Resources on 29-MAY-1996 23:48:06.67
Physical Memory Usage (pages):     Total        Free      In Use    Modified
Main Memory (1536.00Mb)           196608       81046      112498        3064
```

この例では、空きリストに残っている OpenVMS Alpha ページは、わずか 81046 です。◆

4.1.4.1.4 行キャッシュの記憶領域への割当て 行キャッシュは、特定の記憶領域に対応付けられています。1 つ以上の記憶領域の行を 1 つの行キャッシュに格納できますが、1 つの記憶領域がポイントできる行キャッシュは 1 つのみです。次に、行キャッシュを記憶領域に割り当てる例を示します。

```
SQL> ALTER STORAGE AREA
  1> EMPIDS_LOW CACHE USING EMPIDS_LOW_RCACHE;
```

ALTER DATABASE 文または CREATE DATABASE 文の CACHE USING 句を使用すると、データベース内のすべての記憶領域に、デフォルトの行キャッシュを定義できます。

4.1.4.1.5 行キャッシュのテーブルへの割当て 論理領域の行キャッシュは、特定のテーブルやインデックスのすべてのパーティションに対応付けられています。次は、5 つの記憶領域にまたがったパーティションを PARTS テーブルに作成している例です。

```
SQL> CREATE STORAGE MAP INVENT_MAP FOR PARTS
  1> ! --
  2> ! -- Inventory table partitioned by stock number
  3> ! --
  4>  STORE USING (STOCK_ID)
  5>      IN STOCKID_A_E WITH LIMIT OF ('10000')
  6>      IN STOCKID_F_K WITH LIMIT OF ('20000')
  7>      IN STOCKID_L_P WITH LIMIT OF ('30000')
  8>      IN STOCKID_Q_U WITH LIMIT OF ('40000')
  9>      OTHERWISE IN STOCKID_V_Z
 10>      PLACEMENT VIA INDEX STOCK_HASH;
SQL>
   .
   .
   .
```

すべての PARTS テーブルを 1 つの行キャッシュに格納する場合を考えます。テーブルと同じ名前で行キャッシュを定義してください。

```
SQL> ALTER DATABASE FILENAME 'INVENTORY'
  1> ADD CACHE PARTS
  2> ROW LENGTH IS 100 BYTES
  3> CACHE SIZE IS 5000 ROWS;
```

キャッシュ名がテーブル名と同じなので、PARTS テーブルでは、すべてのパーティションのすべての行が自動的にキャッシュされます。

4.1.4.2 メモリー内にキャッシングする内容の制御

ROW REPLACEMENT パラメータによって、行キャッシュがいっぱいになったときに行う処理をある程度制御できます。行キャッシュで行の置換が有効化されている場合、キャッシュがいっぱいになると、新しい行は、最も古く、未使用で、マークが解除されている行と置き換えられます。行の置換が無効化されている場合、キャッシュがいっぱいになると、新しい行はキャッシングされないので、データは常にディスクから取り出されます。

ROW REPLACEMENT パラメータは、メモリー内に行を固定します。次の内容を固定することにより、パフォーマンスを向上できます。

- B ツリー・インデックスのノンリーフ・ノード
  行キャッシュのサイズを指定している場合は、ノード分割を考慮してください。親ノードが分割し、新しいノード用の空き領域がない場合、新しいノードはメモリー内に固定されません。
- 主に読取り専用のデータ
  データ・ウェアハウス環境のファクト表のように、非常に頻繁に変更されるデータは、メモリーに常駐しておくと効果的です。
- データが頻繁に更新され、テーブル全体がキャッシュ内に収まる場合
  キャッシュで置換を定義していない場合、Oracle Rdb は、アクセスを最適化します。

ROW REPLACEMENT IS DISABLED 句を使用すると、データはメモリーに常駐し、その後の読取りは、ディスクではなくメモリーから実行されます。これにより、Oracle Rdb で必要とするロックの数が少なくなるため、パフォーマンスも向上します。

テーブルのアクセス・パターンがランダムな場合、行の置換を有効にすると効果的です。これにより、アクセス頻度が最も高いデータがメモリー内に常駐します。物理メモリー容量の制限で、テーブル全体をキャッシュできない場合も多いので、最も使用頻度の高い行のみをキャッシングすることにより、パフォーマンスが向上します。

4.1.4.2.1 行の置換ストラテジ グローバルおよびローカル・バッファでは、データベース・ページの置換ストラテジとして、最低使用頻度（LRU）を使用します。行キャッシングでは、LRU に変更を加えた置換ストラテジを使用します。各データベース・ユーザーは、最後にアクセスした 10 行を保護できます。この行のグループは、**ワーキング・セット**と呼ばれます。ワーキング・セットに含まれる行は、**参照済**とみなされ、行の置換の対象にはなりません。キャッシュに格納されているが、ワーキング・セットには含まれていない行は、**未参照**の行とみなされます。未参照の行は、置換の対象となります。

4.1.4.3 キャッシュへの行の挿入

各ユーザー・プロセスは、データベースから行を要求します。ユーザー・プロセスは、記憶領域から行を読み取ると、その行をキャッシュに挿入しようとします（キャッシュ内に既に存在していない場合）。使用可能なスロットがあり、十分な空き領域があれば、要求された行はキャッシュに格納されます。キャッシュ内に使用可能なスロットがない場合、次のような処理が行われます。

- ROW REPLACEMENT IS ENABLED を使用し、未参照の行が見つかった場合、未参照の行は、新しい行と置き換わります。Oracle Rdb は、未参照の行をランダムに選択します。
- ROW REPLACEMENT IS DISABLED を使用する場合、行はキャッシュに格納されません。

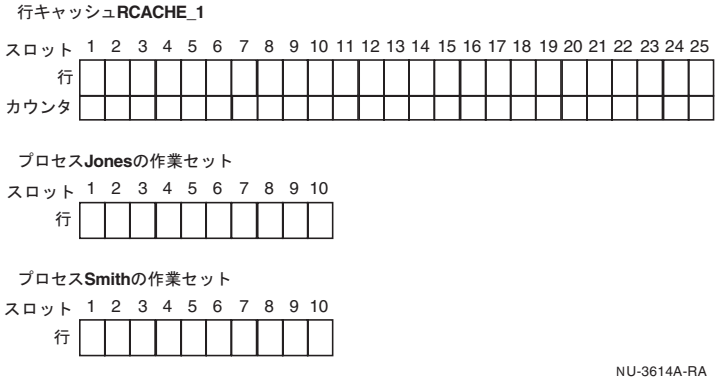
夜間、システムの使用率が低いときに行キャッシュを事前にロードしておけば、日中のデータベース処理がピーク時にも、メモリーからデータを取得できます。

この項の後では、Oracle Rdb がキャッシュに行を挿入する方法について説明します。

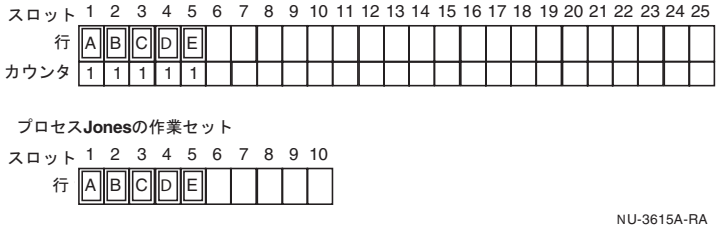
この例では、次のような内容を前提とします。

- 行キャッシュを有効化
- 行の置換を有効化
- 行キャッシュ（RCACHE_1）を 25 スロットで作成
- 2 つのプロセス（Jones と Smith）がデータベースに接続
- 行キャッシュの行は、変更しない

次の図は、最初の割当てを示しています。



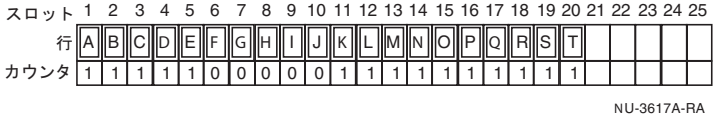
1. プロセス Jones は、5 つの行を、行キャッシュの最初の 5 スロットに読み込むクエリーを実行します。



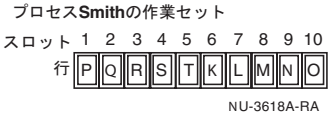
各行スロットは、ワーキング・セット・カウンタを持っています。ワーキング・セット・カウンタは、その行がワーキング・セットに含まれているかどうかを示します。正の値は、その行がワーキング・セットに含まれることを示します。ワーキング・セットに含まれている行は、置換対象にはなりません。

2. プロセス Smith は、データベースから 15 行を要求します。次のように、要求された行のうち、最初の 10 行は、Smith のワーキング・セットに格納されます。

プロセス Smith のワーキング・セットのスロット数は 10 であり、10 スロットが使用済です。アクセス時期が最も古い行が、プロセス Smith がキャッシュに読み込む 11 番目の行に置き換えられます。12 ～ 15 番目の行も、スロット 2 ～ 5 をそれぞれ上書きします。
15 行の読込みが完了すると、キャッシュは次のようになります。

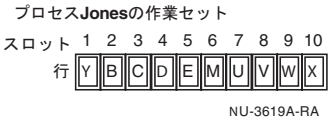


15 行の読込みが完了すると、プロセス Smith のワーキング・セットは次のようになります。

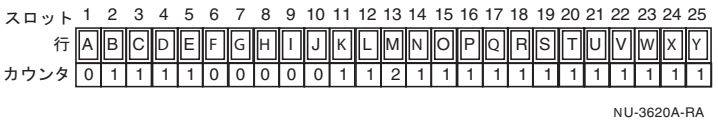


この時点では、行 F、G、H、I、J は未参照です。この行は、キャッシュには存在しますが、どのプロセスのワーキング・セットにも含まれていません。Oracle Rdb は、未参照行のワーキング・セット・カウンタを 0 に設定します。未参照行は、内容が変更されていない状態で、行の置換が有効化されている場合、置換の対象となります。どのプロセスも、I/O を実行しないで行 F、G、H、I、J の読取りを実行できます。ただし、現在キャッシュに存在しない行をプロセスが要求すると、行 F、G、H、I、J のいずれかが新しい行に置き換えられます。
行キャッシュの各スロットは、変更フラグを持っています。行を変更したが、ディスクにはフラッシュしていない状態を、**使用済**の状態と呼びます。使用済行は、行の置換の候補にもなりません。変更された行は、行キャッシュ・サーバー（Row Cache Server：RCS）プロセスがディスクに書き込みます。詳細は、4.1.4.4 項（4-80）を参照してください。

3. プロセス Jones はさらに、M、U、V、W、X、Y、Z の 7 行を要求します。M はキャッシュ内に既に存在するので、Jones は I/O を実行しなくても行 M を読み取れます。行キャッシュのスロットにはデータは格納されませんが、行 M は、プロセス Jones のワーキング・セットに追加されます。
プロセス Jones のワーキング・セットは、次のようになります。



次のように、行 U、V、W、X、Y は、行キャッシュ内の残りのスロットに挿入されます。



スロット 13 のワーキング・セット・カウンタから、行 M が 2 つのワーキング・セットに格納されていることに注意してください。これは、2 つのプロセスが同じ行にアクセスし

ていることを示します。スロットを共有しているプロセスの数は、**共有カウント**と呼ばれます。
この時点で、キャッシュはいっぱいになります。行の置換が無効化されている場合、行 Z は挿入できません。ただし、この例では、行の置換が有効であり、未参照スロットが存在します。したがって、Oracle Rdb は、未参照スロットから置換対象となる行を選び、新しい行である Z を挿入するための空き領域を作ります（この例では、未参照スロットは、A、F、G、H、I、J です）。

4.1.4.4 行キャッシュ・サーバー（Row Cache Server：RCS）プロセス

行キャッシュを有効にすると、Oracle Rdb は、記憶領域ファイルにコミット済の更新を書き込む処理を、**スイープしきい値**と呼ばれるユーザー指定のしきい値まで遅延します。最後のスイープの後で変更されたすべての行が、記憶領域に書き込まれます。行キャッシュに含まれる行のスイープは、**行キャッシュ・サーバー（Row Cache Server：RCS）**プロセスが実行します。

RCS プロセスは、次のような処理を行います。

- マークしたコールド・レコードを行キャッシュからキャッシュ・バックアップ格納域（.rdc）ファイルに書き込む
  RCS プロセスは、チェックポイント間隔が経過すると、キャッシュ・バックアップ格納域ファイルへの書込みを実行します。
- マークしたコールド・レコードを行キャッシュから記憶領域（.rda）ファイルに書き込む
  RCS プロセスは、行キャッシュをスイープしたとき、記憶領域ファイルへの書込みを実行します。

Oracle Rdb は、キャッシュ内の各行に変更フラグを対応付け、行が変更されたかどうかを示します。変更フラグには、次のような値が格納されます。

変更フラグ	意味
Marked Cold（MC）	行は変更されており、次のスイープで .rda ファイルへの書込みが可能。
Marked Hot（MH）	行は変更されている。次のスイープで、行の変更フラグは Marked Cold に変更される。
Unmarked Cold（UC）	行は変更されていない。

RCS は、分離プロセスです。Oracle Rdb は、行キャッシュが有効化されたときに、RCS プロセスを作成します。

4.1.4.4.1 行キャッシュのチェックポイント Oracle Rdb は、各行キャッシュについて、.rdc ファイル拡張子を持ったキャッシュ・バックアップ格納域ファイルを作成します。RCS プロセスは、チェックポイント間隔が経過した時点で、このファイルへの書込みを実行します。

チェックポイント間隔の指定は、次の SQL 構文を使用します。

- CHECKPOINT INTERVAL IS n BLOCKS
  チェックポイント間隔として、キャッシュ・バックアップ格納域ファイルに変更した行を書き込むまでの間、.aij ファイルに蓄積しておくことができるブロック数を指定します。
- CHECKPOINT TIMED EVERY s SECONDS
  チェックポイント間隔として、キャッシュ・バックアップ格納域ファイルに変更した行を書き込むまでの時間を、秒数で指定します。

RCS プロセスでチェックポイントを実行するかどうかは、RDM$BIND_RCS_CHECKPOINT 論理名または RDB_BIND_RCS_CHECKPOINT 構成パラメータで指定します。1 は、チェックポイントを実行し、0 はチェックポイントを実行しません。

デフォルトでは、.rdc ファイルは行キャッシュ・サイズの 40% です。次のように、読取り専用キャッシュには、.rdc ファイルのサイズに 1 ブロックを指定してください。

```
SQL> ALTER DATABASE FILE mf_personnel
  1> ADD CACHE RCACHE_2
  2> LOCATION IS WORK$DISK1:[RCS]
  3> ALLOCATION IS 1 BLOCK;
```

.rdc ファイルの格納先を指定しない場合、デフォルトは、データベース・ルート・ファイル・ディレクトリです。

4.1.4.4.2 行キャッシュのスイープ RCS プロセスには、行キャッシュをスイープするという目的もあります。**スイープ**とは、アクティブな行キャッシュすべてを検査して、変更された行を記憶領域に書き戻す処理です。キャッシュがいっぱいになる前に、RCS が記憶領域に行を書き込むのが理想的です。データベース管理者（DataBase Administrator：DBA）は、次のような内容を指定できます。

- キャッシュにデータがどの程度蓄積された時点で、記憶領域ファイルに変更した行を書き込むか
  これは、スイープしきい値で指定します。
- どの程度の頻度でキャッシュをスイープするか
  これは、スイープ間隔で指定します。

RCS プロセスがスイープを実行中に、変更された行を検出すると、次のような処理が行われます。

- 変更された行が Marked Cold（MC）ならば、変更された行を記憶領域ファイルに書き込み、変更フラグを Unmarked Cold（UC）に変更します。
  変更フラグが Unmarked Cold（UC）になると、スロットは置換の候補になります。

- 変更された行が Marked Hot（MH）ならば、変更フラグを Marked Cold（MC）に変更します。この行は、次のスイープで記憶領域ファイルに書き込まれます。
- フラッシュ・カウンタを 1 だけ加算します。
  Oracle Rdb は、キャッシュ内の行を記憶領域ファイルに書き込んだ回数の追跡情報を記録しています。

RCS プロセスは、作成した順番で、行キャッシュをスイープします。

1 つの行キャッシュには、"start of sweep" と "end of sweep" の 2 つのしきい値が設定されています。変更した行の数が "start of sweep" しきい値を超えると、スイープを開始し、変更した行数が "end of sweep" しきい値を下回るまで継続します。

スイープしきい値の指定は、次の論理名と構成パラメータを使用します。

- RDM$BIND_RCS_MAX_COLD と RDB_BIND_RCS_MAX_COLD
  "start of sweep" しきい値を定義します。
- RDM$BIND_RCS_MIN_COLD と RDB_BIND_RCS_MIN_COLD
  "end of sweep" しきい値を定義します。

OpenVMS VAX OpenVMS Alpha

次の例は、スイープしきい値をそれぞれ 20% と 50% に設定する方法を示しています。

```
$ DEFINE/SYSTEM RDM$BIND_RCS_MIN_COLD 20
$ DEFINE/SYSTEM RDM$BIND_RCS_MAX_COLD 50
$!
$! When the number of modified rows in a row cache exceeds
$! 50%, Oracle Rdb writes the modified rows to the
$! storage area.
$!
$! This sweeping of modified rows occurs until the number of
$! modified rows goes below 20%.
$!
```

これは、システム・レベルの論理名として定義してください。◆

行キャッシュのスイープ間隔は、指定ができます。スイープ間隔とは、RCS スイープの間隔を分単位で示したものです。スイープ間隔を定義するには、RDM$BIND_RCS_SWEEP_INTERVAL 論理名または RDB_BIND_RCS_SWEEP_INTERVAL 構成パラメータを使用します。

OpenVMS VAX OpenVMS Alpha

次の例は、スイープ間隔を 2 分に設定する方法を示しています。

```
$ DEFINE/SYSTEM RDM$BIND_RCS_SWEEP_INTERVAL 2
$!
$! The RCS process takes 2 minutes to write the modified
$! rows to the storage areas. The sweep interval allows
$! the DBA to balance the activity of the RCS process with
$! other users accessing the database.
$!
```

RDM$BIND_RCS_SWEEP_INTERVAL は、システム・レベルの論理名として定義してください。◆

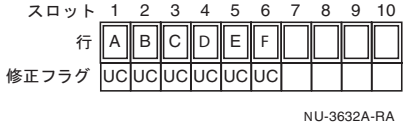
RCS プロセスに対して、しきい値に達していなくてもキャッシュのスイープを開始するよう指示することができます。感嘆符（[!]）を入力し、「Wake up RCS process」オプションを選択して、Performance Monitor のツール機能を起動します。

スイープしきい値やスイープ間隔が適切に設定されていないと、行キャッシュが変更された行でいっぱいになってしまい、RCS プロセスが変更された行を記憶領域ファイルに書き込んで、変更フラグをリセットするまでは、新しい行を挿入できなくなります。キャッシュがいっぱいになっているかどうかを確認するには、Performance Monitor の「Row Cache (One Cache)」画面の "cache full" 統計を参照してください。

行キャッシュが変更された行でいっぱいになる頻度が高い場合、スイープしきい値とスイープ間隔を調整してください。Performance Monitor の RCS Dashboard を使えば、スイープしきい値とスイープ間隔に異なる値を設定し、テストできます。適切な値が決まったら、該当する論理名や構成パラメータを定義してください。

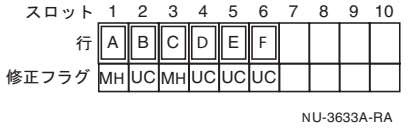
この項の残りの部分では、RCS プロセスのスイープで、どのような処理が実行されるか説明します。

1. プロセス 1 がクエリーを実行した結果、6 行がキャッシュに格納されています。



   この時点では、キャッシュ内のすべてのレコードは、変更されていないので、"Unmarked Cold（UC）" となります。

2. プロセス 2 は、スロット 1 と 3 の行を更新します。

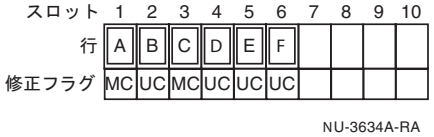


   スロット 1 と 3 は、"Marked Hot（MH）" になります。

3. RCS プロセスは、次のような方法で、スイープを開始します。

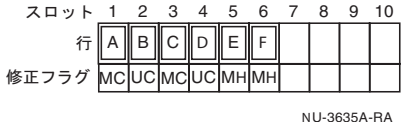
   - "Marked Cold（MC）" については、行を記憶領域に書き込み、変更フラグを "Unmarked Cold（UC）" に変更します。

- "Marked Hot（MH）" については、変更フラグを "Marked Cold（MC）" に変更します。



RCS プロセスは、最初のスイープでは、変更された行があっても記憶領域への書込みは行いません。行が変更された場合、RCS は、変更フラグを "Unmarked Cold（UC）" から "Marked Hot（MH）" に変更します。

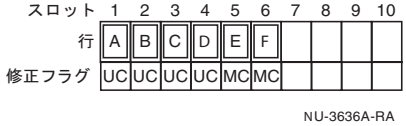
4. プロセス 2 は、クエリーを実行した結果、スロット 5 と 6 の行を更新します。



5. RCS プロセスは、次のような方法で、スイープを開始します。

- "Marked Cold（MC）" については、行を記憶領域に書き込み、変更フラグを "Marked Cold（MC）" から "Unmarked Cold（UC）" に変更します。
- "Marked Hot（MH）" については、変更フラグを "Marked Cold（MC）" に変更します。

スイープが完了すると、キャッシュは次のようになります。



スイープの間、スロット 1 と 3 を記憶領域ファイルに書き込み、スロット 1 と 3 の変更フラグを "Marked Cold（MC）" から "Unmarked Cold（UC）" に変更します。RCS プロセスは、次のスイープで、スロット 5 と 6 の行を記憶領域ファイルに書き込みます。

4.1.4.5 物理および論理領域キャッシュの使用

キャッシュは、物理領域キャッシュと論理領域キャッシュの両方を使用できます。たとえば、ハッシュ・インデックス内のレコードは、次のようなサイズであると仮定します。

- システム・レコード ---16 バイト（STOCK_SYS キャッシュ）
- ハッシュ・バケット・レコード ---100 バイト（STOCK_HASH キャッシュ）
- データ・レコード ---320 バイト（STOCK キャッシュ）

3 つのレコード・タイプすべてに対してキャッシュを 1 つ作成し、行サイズに 320 バイトを指定すると、システム・レコードとハッシュ・バケット・レコードの格納で、大量のメモリーが無駄になってしまいます。レコード・タイプ別にキャッシュを 3 つ作成する方が効率的です。

例 4-12（4-85）は、行キャッシュの定義です。

例 4-12 行キャッシュの定義

```
SQL> --
SQL> -- The following cache definition is for the system records:
SQL> --
  1>  ADD CACHE STOCK_SYS
  2>      CACHE SIZE IS 5000 rows
  3>      ROW LENGTH IS 16 bytes
  4>      NUMBER OF RESERVED ROWS IS 1200
  5>      SHARED MEMORY IS SYSTEM
  6>      LARGE MEMORY IS ENABLED
  7>      ROW REPLACEMENT IS ENABLED
  8>      LOCATION IS '$1$DKA400:[RCS]'
  9>      ALLOCATION IS 1 BLOCK
 10>      EXTENT IS 1 BLOCK
 11>      NUMBER OF SWEEP ROWS IS 1000;
SQL> --
SQL> -- The following cache definition is for the hash bucket:
SQL> --
  1>  ADD CACHE STOCK_HASH
  2>      CACHE SIZE IS 5000 rows
  3>      ROW LENGTH IS 100 BYTES
  4>      NUMBER OF RESERVED ROWS IS 1200
  5>      SHARED MEMORY IS SYSTEM
  6>      LARGE MEMORY IS ENABLED
  7>      ROW REPLACEMENT IS ENABLED
  8>      LOCATION IS '$1$DKA400:[RCS]'
  9>      ALLOCATION IS 1 BLOCK
 10>      EXTENT IS 1 BLOCK
 11>      NUMBER OF SWEEP ROWS IS 1000;
SQL>
```

```
SQL> --
SQL> -- The following cache definition is for the data records:
SQL> --
  1>  ADD CACHE STOCK
  2>      CACHE SIZE IS 5000 rows
  3>      ROW LENGTH IS 320 BYTES
  4>      NUMBER OF RESERVED ROWS IS 2400
  5>      SHARED MEMORY IS SYSTEM
  6>      LARGE MEMORY IS ENABLED
  7>      ROW REPLACEMENT IS ENABLED
  8>      LOCATION IS '$1$DKA400:[RCS]'
  9>      ALLOCATION IS 1750000 BLOCKS
 10>      EXTENT IS 1500000 BLOCK
 11>      NUMBER OF SWEEP ROWS IS 1000;
SQL>
```

3 つのレコード・タイプすべてにキャッシュを 1 つ使用する場合は、最長の行にも十分に対応できる行サイズを確保してください。例 4-12（4-85）では、データ行は 320 バイトです。オーバーヘッド分に 10 バイトを加算すると、1 つの行キャッシュが使用するメモリー容量は、次のように計算できます。

```
Total number of bytes
          = (# of rows in cache × row length of largest row)
          = (15000 × 330)
          = 4950000 bytes
```

3 つの行キャッシュが使用するメモリー容量は、次のように計算できます。

```
Total number of bytes
          = (# of rows in cache × row length of system record) +
            (# of rows in cache × row length of hash bucket) +
            (# of rows in cache × row length of data record)
          = (5000 × 16) +
            (5000 × 100) +
            (5000 × 330)
          = 1735000 bytes
```

この方が、かなりの容量のメモリーを節約できます。

4.1.4.6 Performance Monitor 画面と行キャッシュ

Performance Monitor の次の画面では、行キャッシュのパフォーマンスと効率に関する統計を表示できます。

- Summary Cache Statistics
- Summary Cache Unmark Statistics
- Row Cache (One Cache)

- Row Cache (One Field)
- Row Cache Utilization
- Hot Row Information
- Row Cache Status
- Row Cache Queue Length
- Row Length Distribution
- RCS Statistics
- Row Cache Dashboard
- RCS Dashboard
- Per-Process Row Cache Dashboard

各画面の詳細は、Performance Monitor のヘルプを参照してください。この項の残りの部分では、行キャッシュに関する画面の例を掲載し、説明します。

4.1.4.6.1 「Row Cache (One Cache)」画面 次の例は、行長が適切なサイズに設定されていないため、行をキャッシュに格納できない状態を示しています。

```
Node: BONZAI          Oracle Rdb V7.0-00 Performance Monitor 11-APR-1996 16:24:57
Rate: 3.00 Seconds            Row Cache (EMPIDS_LOW)          Elapsed: 00:02:54.98
Page: 1 of 1       RDBVMS_USER1:[LOGAN.V70]MF_PERSONNEL.RDB;1       Mode: Online
-------------------------------------------------------------------------------
statistic..... rate.per.second..... total..... average.....
name.......... max..... cur..... avg..... count..... per.trans....
latch requests              0        0        0.0            0           0.0
  retried                   0        0        0.0            0           0.0
cache searches            536      530      507.5        27066       27066.0
  found in workset          0        0        0.0            0           0.0
  found in cache            0        0        0.0            0           0.0
  found too big             0        0        0.0            0           0.0
insert cache              536      530      507.5        27066       27066.0
  row too big             536      530      507.5        27066       27066.0
  cache full                0        0        0.0            0           0.0
  collision                 0        0        0.0            0           0.0
VLM requests                0        0        0.0            0           0.0
  window turns              0        0        0.0            0           0.0
skipped dirty slot          0        0        0.0            0           0.0
skipped inuse slot          0        0        0.0            0           0.0
hash misses                 0        0        0.0            0           0.0
cache unmark                0        0        0.0            0           0.0
-------------------------------------------------------------------------------
Exit Graph Help Menu Options Reset Set_rate Time_plot Unreset Write X_plot !
```

次の例は、要求された行がすべてキャッシュに存在している状態を示しています。

```
Node: BONZAI         Oracle Rdb V7.0-00 Performance Monitor 11-APR-1996 16:28:03
Rate: 3.00 Seconds           Row Cache (EMPIDS_MID)         Elapsed: 00:02:58.81
Page: 1 of 1       RDBVMS_USER1:[LOGAN.V70]MF_PERSONNEL.RDB;1       Mode: Online
------------------------------------------------------------------------------
statistic........... rate.per.second.............. total....... average......
name................ max..... cur..... avg....... count....... per.trans....
latch requests              0        0        0.3         4763         297.6
  retried                   0        0        0.0            0           0.0
cache searches           2585       95       10.3       151297        9456.0
  found in workset          0        0        0.0            0           0.0
  found in cache         2585       95        3.4        51297        3206.0
  found too big             0        0        0.0            0           0.0
insert cache                6        0        6.8       100000        6250.0
  row too big               0        0        0.0            0           0.0
  cache full                0        0        0.0            0           0.0
  collision                 0        0        0.0            0           0.0
VLM requests                0        0        0.0            0           0.0
  window turns              0        0        0.0            0           0.0
skipped dirty slot          0        0        0.0            0           0.0
skipped inuse slot          0        0        0.0            0           0.0
hash misses               820        0        2.4        36237        2264.8
cache unmark                0        0        0.0            0           0.0
------------------------------------------------------------------------------
Exit Graph Help Menu Options Reset Set_rate Time_plot Unreset Write X_plot !
```

"found in workset" 統計と "found in cache" 統計は、キャッシュの効率を示しています。統計の値が 100% に近くなるほど、行キャッシュが効率的に使用されていることを示します。

次の例でのキャッシュ・ヒット率は 75% です。

```
Node: BONZAI         Oracle Rdb V7.0-00 Performance Monitor 11-APR-1996 16:33:03
Rate: 3.00 Seconds           Row Cache (EMPIDS_MID)         Elapsed: 00:03:03.10
Page: 1 of 1       RDBVMS_USER1:[LOGAN.V70]MF_PERSONNEL.RDB;1       Mode: Online
------------------------------------------------------------------------------
statistic........... rate.per.second.............. total....... average......
name................ max..... cur..... avg....... count....... per.trans....
latch requests             18       16        0.3        11143           0.3
  retried                   0        0        0.0            0           0.0
cache searches           2914     1958       51.4      1501158          50.0
  found in workset        355      225        5.2       154416           5.1
  found in cache         1851      883       19.3       564795          18.8
  found too big             0        0        0.0            0           0.0
insert cache              721      608       14.6       429294          14.3
  row too big               0        0        0.0            0           0.0
  cache full               58       58        0.7        22839           0.7
  collision                 0        0        0.0            3           0.0
```

```
VLM requests                1957     1041       18.2       533613          17.7
  window turns               117       33        1.1        34495           1.1
skipped dirty slot             0        0        0.0            0           0.0
skipped inuse slot             3        0        3.8       112321           3.7
hash misses                  391      391        5.5       162170           5.4
cache unmark                 235      116        3.1        91713           3.0
------------------------------------------------------------------------------
Exit Graph Help Menu Options Reset Set_rate Time_plot Unreset Write X_plot !
```

次は、「RCS Dashboard」画面の例です。

```
Node: BONZAI          Oracle Rdb V7.0-00 Performance Monitor 11-APR-1996 16:34:00
Rate: 3.00 Seconds               RCS Dashboard              Elapsed: 00:11:57.85
Page: 1 of 1       RDBVMS_USER1:[LOGAN.V70]MF_PERSONNEL.RDB;1       Mode: Online
------------------------------------------------------------------------------
Database.......... Current... Previous.. Lowest.... Highest... Original.. Chng
Attribute.Name.... Value..... Value..... Value..... Value..... Value..... Cnt.
Ckpt Buffer Count          15         15         15         15         15    0
Batch Count             10000      10000      10000      10000      10000    0
Checkpoint                  1          1          1          1          1    0
Ckpt Time Interval        300        300        300        300        300    0
Sweep Interval              1          1          1          1          1    0
Low Cold Thshld       2899000    2899000    2899000    2899000    2899000    0
High Cold Thshld      2900000    2900000    2900000    2900000    2900000    0
Cold Record Count           0          0          0          0          0    0
Abort Sweep                 1          1          1          1          1    0
------------------------------------------------------------------------------
Config Exit Help Menu Options Set_rate Write !
```

4.1.3.1 項（4-62）で説明した条件が 1 つでも満たされていないと、Performance Monitor のメイン・メニューには、行キャッシュの画面は表示されません。

4.1.5 高速コミット・トランザクション処理

Oracle Rdb が更新されたページをメモリー・バッファからディスクへ書き込む方法とタイミングを指定することにより、データベース・パフォーマンスを向上できる場合があります。

- デフォルトでは、Oracle Rdb は、トランザクションが COMMIT 文を実行するたびに、更新されたデータベース・ページをディスクへ書き込みます。コミットの前にトランザクションに障害が発生した場合、Oracle Rdb で必要な処理は、現在の障害トランザクションのロールバックのみです。前に処理が成功しているトランザクションを再処理する必要はありません。
- **高速コミット**トランザクション処理（4.1.5.4 項（4-100）を参照）を有効にすると、Oracle Rdb は、更新されたページをバッファ・プール内に保持し、トランザクションのコミットが発生しても、ページを書き込みません。ユーザー指定のしきい値（チェックポイントと呼ばれます。4.1.5.2 項（4-93）を参照）に達するまで、更新されたページを

バッファ・プール内に格納しておくことができ、しきい値に達すると、複数のトランザクションで更新されたページをディスクにフラッシュします。トランザクションに障害が発生した場合、Oracle Rdb は、現在の障害トランザクションをロールバックし、最後のチェックポイントの後にコミットされたトランザクションをすべて再処理する必要があります。

高速コミット・トランザクション処理は、削除、更新、挿入などのデータ更新のみに適用されます。論理領域の作成やインデックスの作成など、データ定義文を含むトランザクションでは、トランザクションの最後で強制的にチェックポイントが発生します。

注意：高速コミット処理を使用するには、After-image ジャーナル（AIJ）を有効にしてください。

この項の残りでは、デフォルトのコミット処理について説明します。高速コミット処理の詳細は、4.1.5.1 項（4-91）を参照してください。

デフォルトのコミット処理では、Oracle Rdb は、次の手順でトランザクション（ここでは、T1 とします）を更新します。

1. トランザクション T1 を開始します。
2. データベース・ページをユーザーのバッファに読み込みます。
3. ページを更新します。
   アプリケーションまたはユーザーが COMMIT 文を発行すると、次のような処理を行います。
4. RUJ バッファを .ruj ファイルにフラッシュします。
5. 更新されたデータベース・ページをディスクにフラッシュします。
6. AIJ バッファを .aij ファイルにフラッシュします（After-image ジャーナルが有効の場合）。
7. T1 のコミット情報をデータベース・ルート・ファイルに書き込みます。
8. 次のトランザクション T2 を開始します。

トランザクション T2 に障害が発生すると、前にコミットされているトランザクション T1 の更新ページやコミット情報は既にディスクに書き込まれているので、T1 はリカバリの対象になりません。Oracle Rdb は、失敗したトランザクション T2 と .ruj ファイルの情報をロールバックします。

デフォルトのコミット処理のリカバリでは、障害が起きたトランザクションに関連する処理のみを再処理するだけでよいので、リカバリは非常に迅速です。この方法では、コミットの I/O 時間（RUJ バッファを .ruj ファイルへフラッシュ、更新ページをディスクへフラッシュ、データベース・ルート・ファイルへの書込み）が必要になりますが、リカバリ時間を最適化できます。

次の項目の一部またはすべてがユーザー環境に当てはまる場合は、デフォルトのトランザクション処理を使用してください。

- システム障害や異常停止したプロセスがある（[Ctl] キーを押しながら [Y] キーを押す）場合。デフォルトのコミット処理は、リカバリ時間を最適化します。
- 次のようなトランザクション・パターンがある場合。
  - 大量のページを更新する結果、バッファ・プールがオーバーフローする
  - 複数の更新トランザクションで大量のページを共有するため、トランザクション間でページのスワップが発生する
  - 同じページの更新を反復して実行しないトランザクション（1 つのトランザクションまたは複数のトランザクション間）

  最初の 2 つのパターンでは、高速コミットの有効と無効に関係なく、更新ページはディスクに書き込まれます。3 つ目のパターンでは、ページの更新は 1 回のみなので、ページをバッファ・プールに格納する必要はありません。
  上記のパターンは、デフォルトのコミット方法を使う絶対的な基準ではありません。厳格なガイドラインはないので処理方法の変更は可能ですが、変更の前後には、データベース・パフォーマンスが最適化されていることを確認してください。

4.1.5.1 高速コミット処理の方法

高速コミット処理を有効にすると、Oracle Rdb は、次の手順でトランザクション（ここでは、T1 とします）を更新します。

1. トランザクション T1 を開始します。
2. データベース・ページをユーザーのバッファに読み込みます。
3. ページを更新しますが、アプリケーションまたはユーザーが COMMIT 文を発行すると、RUJ バッファの .ruj ファイルへのフラッシュや、更新したデータベース・ページのディスクへのフラッシュは行いません。
4. AIJ バッファを .aij ファイルにフラッシュします。After-image ジャーナルを有効化する必要があります。
5. T1 のコミット情報をデータベース・ルート・ファイルに書き込みます。高速コミットを有効化した状態で JOURNAL OPTIMIZATION オプションを有効化すると、この手順は省略できます。詳細は、4.1.5.3 項（4-98）を参照してください。
6. 次のトランザクション T2 を開始します。
7. チェックポイントに達したら、更新ページをフラッシュし、プロセスはデータベースから切断されるか、メタデータの変更が実行されます。

トランザクションに障害が発生した場合、最後のチェックポイント以降に発生したトランザクションについては、更新されたページをディスクに書き込んでないので、これらのトランザクションはすべて再処理する必要があります。Oracle Rdb は、.aij ファイルに書き込まれている情報を使用して、トランザクションを再処理します。また、（バッファ・プール・オーバーフローによって）データベース・ファイルへの書き込みが完了しているため、RUJ バッファやファイルに情報が格納されているトランザクションをロールバックする必要があります。

高速コミット処理では、トランザクションのコミット時間を短縮できますが、リカバリ処理時間は長くなります。最後のチェックポイントの後で更新されたトランザクションが複数ある状態で障害が発生すると、チェックポイント後のトランザクションをすべて再処理しなくてはならないからです。ただし、チェックポイント間隔を短くすることにより、リカバリ時間を短縮できます（4.1.5.2 項（4-93）を参照してください）。

トランザクション処理環境が安定しており、次のような条件が当てはまる場合は、高速トランザクション処理を有効化してください。

- プロセスが、複数のトランザクションで同じ行を更新する
  たとえば、パーツのデータベースなどでは、受注のたびに在庫を調整し、在庫を補充します。デフォルトのコミット処理では、1 つのトランザクションの最後に、更新された行を含むページをディスクにフラッシュします。したがって、ディスク書込みが大量に発生します。高速コミット処理を有効化すると、更新された行を含むページはプロセスのバッファ・プールに保持され、ディスクへの書込みは、チェックポイント時のみに実行されます。更新ページをメモリー内に保持することにより、ページ I/O を減らせるだけでなく、トランザクションごとに RUJ バッファを .ruj ファイルにフラッシュする必要がないので、RUJ I/O の数も減少します。
- トランザクションは短く、更新するページの数が少ない
  この条件では、更新されたページをトランザクションごとにフラッシュするのではなく、バッファ・プールに保持しておいて、チェックポイントですべてをフラッシュする方が効率的です。たとえば、一度に 10 のデータ I/O を（チェックポイント時に）発行する方が、同じ 10 の I/O を 1 つずつ（コミットのたびに）発行するよりも効率的です。また、RUI I/O 操作の数も減らすことができます。

表 4-6（4-93）は、高速コミット・トランザクション処理のガイドラインの概要です。

表 4-6 高速コミット処理の有効化と無効化の基準

トランザクション・タイプ	安定した環境	不安定な環境
同じ行を繰り返し更新	有効化	可 [*1]
トランザクションが短く、更新するページが少ない	有効化	可 [*1]
バッファ・プール・オーバーフローで大量のページを更新	無効化	無効化
複数のトランザクションが同じページを共有	無効化	無効化
同じ行を頻繁に更新しない	無効化	無効化

[*1] 異常終了が発生する環境で高速コミット処理を有効化すると、デフォルトのコミット方法よりもリカバリ処理時間が長くなります。高速コミット処理を使用する場合は、障害の発生頻度と、許容できるリカバリ時間を検討してください。

高速コミットを有効にした場合、環境が安定しているかどうかによって、スナップショットの書込みの有効と無効が決まります。システムやプロセスの障害によるデータベースのリカバリが頻繁に発生しない場合は、スナップショットを即時または遅延で有効にしてください。高速コミットを有効にしている場合のリカバリ処理では、最後のチェックポイント以降にコミットしたすべてのトランザクションについて、スナップショット・ファイルへのレコード書込みとランザクションの再処理を実行しなければならないので、リカバリが頻繁に必要になる環境では、スナップショットの書込みを無効にすることもできます。スナップショットを有効にすると、リカバリ時間は長くなります。ただし、ジャーナルの最適化オプションを有効にする場合は、スナップショットを無効または遅延する必要があります。

4.1.5.2 チェックポイント処理

高速コミット処理を有効にすると、プロセスは、トランザクションのコミットのたびに更新ページをフラッシュしません。プロセスに障害が発生すると（メモリーの変更内容が失われる）、データベース・リカバリ（DataBase Recovery：DBR）プロセスは、プロセスがすでに更新したトランザクションを再処理し、中断したトランザクションが変更した内容をロールバックしなくてはなりません。更新内容の再処理では、.aij ファイルを読み取り、データ・ページへの変更内容を再適用します。

DBR が再処理するトランザクションの数の上限は、チェックポイント間隔で設定できます。チェックポイント間隔を設定することにより、Oracle Rdb は、更新済ページを定期的にフラッシュします。この値によって、DBR が再処理しなくてはならないトランザクションの数が制限されるので、リカバリ時間を短縮できます。チェックポイント間隔を設定しないと、たとえば、プロセスが 1000 件のトランザクションを完了した状態で、1001 件目に障害が発生した場合、DBR は、1 ～ 1000 件のトランザクションを再処理し、1001 件目をロールバックしなくてはなりません。

注意： DBR のリカバリでは、DBR は、他のプロセスがデータベースにアクセスできないように、データベース全体にフリーズ・ロックを取得します。これは、デフォルトのコミット処理でも発生します。ただし、高速コミットを有効化すると、複数のトランザクションを再処理するので、一般的に DBR による処理は増大します。したがって、高速コミットを有効にすると、DBR がフリーズ・ロックを取得する時間は長くなります。次は、高速コミットを有効にすると、追加で必要になる処理内容の一覧です。

最後のチェックポイント以降の .aij ファイルの読取り	高速コミット
ディスクにフラッシュしていないコミット済トランザクションの再処理	高速コミット
.ruj ファイルの読取り	高速コミットとデフォルト・コミット
RUJ 情報を元にした変更内容のロールバック	高速コミットとデフォルト・コミット

高速コミット・トランザクション処理は、AIJ バックアップ・プロシージャで重要な役割を果たします。詳細は、『Oracle Rdb7 Guide to Database Maintenance』を参照してください。

高速コミットを有効化する場合、次の方法の一方または両方を使用して、データベース全体でチェックポイント間隔を指定できます。

- 更新ページをフラッシュするまでに .aij ファイルに蓄積しておけるブロックの数を指定します。すべてのプロセスが、.aij ファイルへの書込みを行います。
- チェックポイント間の経過時間を指定します。

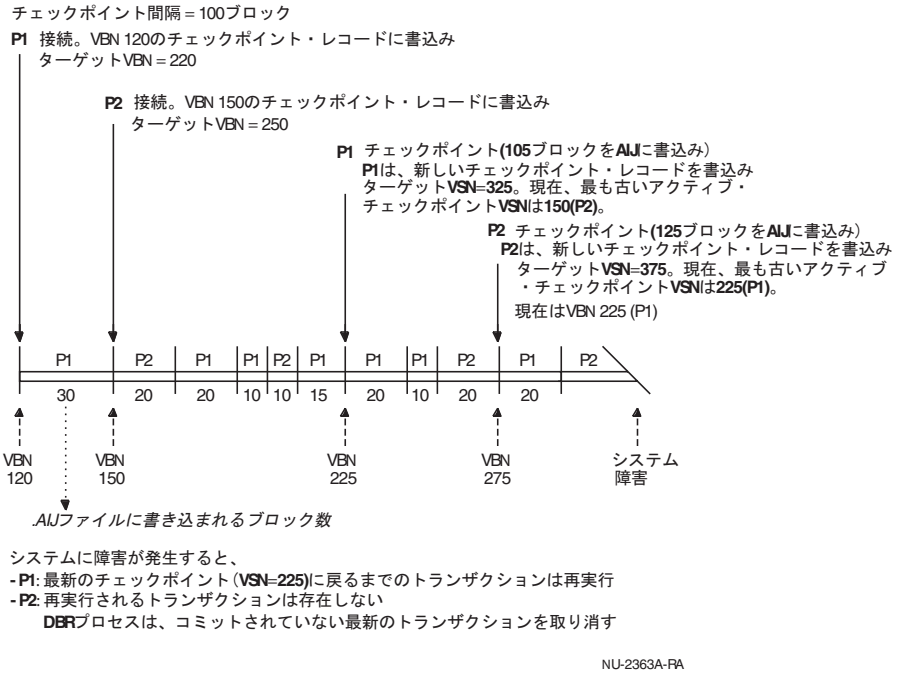
たとえば、チェックポイント間隔を 100 ブロックに設定すると、すべてのプロセスは、最後のチェックポイントから 100 ブロックが .aij ファイルに書き込まれた時点で、更新されたページをディスクにフラッシュします。また、チェックポイント間隔を 90 秒に設定すると、すべてのプロセスは、最後のチェックポイントから 90 秒経過した時点で、更新されたページをディスクにフラッシュします。高速コミットの構文の詳細は、4.1.5.4 項（4-100）を参照してください。

プロセスは、データベースに接続すると、チェックポイント・レコードを .aij ファイルに書き込み、チェックポイント・レコードの位置を示す .aij ファイルの仮想ブロック番号（Virtual Block Number：VBN）を記録します。チェックポイントの VBN が 120、チェックポイント間隔が 100 ブロックの場合、プロセスは、VBN が 220 に達した時点で、チェックポイント処理を行います。図 4-9（4-95）を参照してください。すべてのプロセスが .aij ファ

イルへの書込みを行うので、更新をあまり行わないプロセスよりも、更新を頻繁に実行するプロセスの方が、チェックポイントまでにコミットするトランザクション数は多くなります。逆に、あるプロセスの最初に長いトランザクションあり、他のプロセスは大量の更新を行う場合、最初のトランザクションをコミットした時点で、チェックポイントが発生する可能性もあります。次に、チェックポイントをまとめます。

- チェックポイント間隔は、プロセスが変更したページをディスクにフラッシュするまでの間、.aij ファイルに蓄積しておくことができるブロック数や経過時間です。
- チェックポイント間隔は、データベースに接続しているすべてのプロセスに適用されます。
- すべてのプロセスが、.aij ファイルへの書込みを行います。
- チェックポイント（更新内容をディスクへフラッシュ）は、トランザクション単位で発生するものであり、トランザクションの途中では発生しません。

図 4-9 チェックポイント処理



データベースのチェックポイント間隔に指定した値に達すると、Oracle Rdb は、次の手順を実行します。

1. 更新されたページをディスクにフラッシュします。
2. チェックポイント・レコードを .aij ファイルに書き込みます。
3. 各プロセスについて、データベース・ルート・ファイルを更新し、チェックポイント・レコードを .aij ファイルに格納したことを記録します。DBR は、ルート・ファイルに格納されているチェックポイント情報に基づいて、.aij ファイルのリカバリを開始するかどうかを判断します。

チェックポイント間隔の設定では、次のような点を考慮してください。

- チェックポイント間隔が小さすぎると、Oracle Rdb は頻繁にフラッシュすることになるので、高速コミット処理を有効にする利点がなくなります。
- 間隔を大きくすると、プロセスに障害が発生した場合のリカバリ時間や再処理の時間が長くなります。
- チェックポイント間隔に経過時間を指定する場合は、トランザクション・レートと、1 件のトランザクションで更新されるブロック数を把握しておいてください。.aij ファイルの伸長の指定とは異なるので、これがわからないと、障害時にリカバリが必要なブロック数やリカバリ時間がわかりません。

障害時に、DBR プロセスがどの程度の .aij ファイル・サイズを再処理できるかを考えてください。各 AIJ チェックポイント間隔で必要になるリカバリ時間の合計は、モニター・ログ・ファイルから推定できます。リカバリの開始と完了を示すログ・ファイル・エントリを 2 つ探してください。この 2 つのエントリの時間の差が、リカバリ時間の推定値です。この値をもとに、AIJ チェックポイント間隔を調整してください。リカバリ時間を短縮したい場合は、チェックポイント間隔を短くします。リカバリ時間が長くてよい場合は、間隔を長くします。

ここでは、1 日 24 時間体制で電話注文を受ける会社の例を使って、データベース管理者（DBA）が、どのチェックポイント・オプションを使用すればよいかを判断する方法を説明します。データベース・ユーザーであるオペレータは、注文を受けて、会社のデータベースに情報を入力します。

日中、電話で大量の注文を受けます。注文の件数に伴って、.aij ファイルのサイズは急増します。DBA は、次の点を把握しています。

- 日勤のオペレータは 100 人
- 1 時間あたりの平均注文件数は、オペレータ 1 人あたり 10 件
- 1 件の注文（データベース・トランザクションに相当）の入力にかかる時間は 6 分
- トランザクション 1 件で、.aij ファイルに 1 ブロックが追加
- システム全体のトランザクションは、1 時間あたり 1000 件
- .aij ファイルは、1 時間あたり 1000 ブロック増加

日中は、1 時間に 1000 ブロックと、.aij ファイルのサイズは急激に増加するので、更新されたページがディスクに書き込まれる時点では、.aij ファイルは大きくなりすぎています。DBA は、このような処理方法は適切ではないと考えています。プロセスに障害が発生すると、DBR は、最後のチェックポイント以降に発生したすべてのトランザクションを再処理しなくてはなりません。チェックポイント以降に発生したトランザクションの件数が多くなるほど、リカバリ時間は長くなります。チェックポイント間隔が小さすぎると、Oracle Rdb は頻繁に更新内容を書き込むことになるので、高速コミット処理を有効にする利点がなくなります。間隔を大きくすると、プロセスに障害が発生した場合のリカバリ時間や再処理の時間が長くなります。

DBA は、チェックポイント間隔を 1000 ブロックに設定します。その結果、オペレータ 1 人あたり 1 時間に 1 回チェックポイントが発生することになります。

ただし、夜間は、システムの使用率は大幅に軽減されます。DBA は、次の点を把握しています。

- 夜勤のオペレータは 2 人
- 1 時間あたりの平均注文件数は、オペレータ 1 人あたり 10 件
- 1 件の注文（データベース・トランザクションに相当）の入力にかかる時間は 6 分
- トランザクション 1 件で、.aij ファイルに 1 ブロックが追加
- 日中は、1 時間あたり 1000 件のトランザクションが発生するのに対して、夜間は 20 件のみ
- .aij ファイル・サイズの増加は、1 時間あたり 20 ブロック

夜間は .aij ファイル・サイズの増加が遅いので、チェックポイントである 1000 ブロックに達しないことがわかりました。もちろん、障害が発生すれば、.aij ファイルは小さいので、リカバリ時間も短くなります。DBA は、安全策として、夜間は、更新ページのディスクへの書込みを少なくとも数回は実行したいと考えています。ただし、チェックポイント間隔オプションの AIJ ブロックは、日中の処理状態に合わせて設定されているので、変更したくありません。DBA は、間隔オプションを使って、2 時間ごとにチェックポイントを実行することにしました。

間隔のチェックポイントを設定しても、チェックポイント間隔のブロック数には影響しませんが、夜勤用の 2 時間ごとのチェックポイントが追加されます。

一般的に、AIJ ブロック・サイズは、リカバリにかかる時間を最も明確に示しているので、重要なチェックポイント・オプションです。ただし、システムの使用パターンや、AIJ ブロック・サイズがリカバリ時間におよぼす影響がよくわからない場合は、間隔オプションを使うと便利です。

チェックポイント間隔の値は、DBA が設定し、データベースに接続しているすべてのプロセスに適用されます。ユーザーは、論理名 RDM$BIND_CKPT_TRANS_INTERVAL または構成パラメータ RDB_BIND_CKPT_TRANS_INTERVAL を定義することにより、プロセス固

有のチェックポイントを別に使用できます。この方法では、チェックポイントに、トランザクション・カウントを使用します。たとえば、ユーザーが RDM$BIND_CKPT_TRANS_INTERVAL または RDB_BIND_CKPT_TRANS_INTERVAL を 10 に定義した場合、チェックポイント間隔の値に達していなくても、トランザクション・カウントの上限である 10 件に達した時点で、更新されたページはディスクにフラッシュされます。3 つの条件（.aij ファイルの伸長、経過時間、トランザクション数）のいずれかが満たされれば、チェックポイントはアクティブになります。チェックポイントが 1 つ発生すると、すべての条件がリセットされます。高速コミットを無効にした場合、RDM$BIND_CKPT_TRANS_INTERVAL 論理名と RDB_BIND_CKPT_TRANS_INTERVAL 構成パラメータは無視されます。

RDM$BIND_CKPT_TRANS_INTERVAL または RDB_BIND_CKPT_TRANS_INTERVAL を定義するときは、トランザクション・パターンをよく把握しておいてください。たとえば、1 つの注文の処理に 5 件のトランザクションが実行され、各トランザクションでは一部のデータを更新することはあっても、次の注文で前の注文データを更新することがほとんどないということがわかっていれば、RDM$BIND_CKPT_TRANS_INTERVAL または RDB_BIND_CKPT_TRANS_INTERVAL を 5 に設定できます。

すべてのノードのデータベースでアクティブになっている全プロセスについて、チェックポイント処理をすぐに実行したい場合は、例 4-13（4-98）のように、RMU Checkpoint コマンドを実行します。

例 4-13 すべてのアクティブなデータベース・プロセスでのチェックポイント処理の強制実行

```
$ RMU/CHECKPOINT mf_personnel
$ rmu -checkpoint mf_personnel
```

このコマンドは、変更されたデータベースのキャッシュ・バッファをすべてディスクにフラッシュします。チェックポイント処理により、DBR による再処理のパフォーマンスも向上します。ただし、パフォーマンスを考慮して、プロセスごとのパラメータを適切に初期化しておく必要があります。すべてのアクティブなデータベース・プロセスが、チェックポイント処理を問題なく実行すると、コマンドは完了します。RMU Checkpoint コマンドの詳細は、『Oracle RMU Reference Manual』を参照してください。

4.1.5.3 ジャーナルの最適化オプション

高速コミット・トランザクション処理を有効にした場合は、ジャーナルの最適化を有効にしてください。このオプションは、データベース・ルートに対する I/O の大部分を省略することにより、コミット処理を大幅に高速化できます。このオプションを使うと、更新集中型のデータベース環境でのパフォーマンスを向上できます。ジャーナルの最適化オプションを有効にする場合には前提条件があるので（この項の後で説明）、汎用データベースや、読取り専用トランザクションが大半を占めるデータベースでは効果を発揮しません。

デフォルトでは、1 件のトランザクションをコミットすると、.ruj ファイルへの書込み、更新されたページのディスクへの書込み、.aij ファイルへの書込み、コミット情報のルート・ファイルへの書込みなどが実行されます。ルート・ファイルの書込みは、アプリケーションが 1

秒あたりに処理可能な最大トランザクション数が減少する原因となるため、パフォーマンスやトランザクション・スループットの高いアプリケーションではボトルネックになります。

高速コミット・リカバリ処理を有効にすると、.ruj ファイルとディスクへの書込みを行いません。また、ジャーナルを最適化すると、ルート・ファイルへの書込みで発生するオーバーヘッドをほとんどなくすことができます。高速コミットとジャーナルの最適化を組み合せることにより、トランザクション処理速度を大幅に向上できます。

ジャーナルの最適化を有効にするには、次のような前提条件があります。

- 高速コミット処理を有効にします。これにより、リカバリが必要な場合、チェックポイント情報を使ってデータベース・リカバリ処理を行えます。
- After-image ジャーナルを有効にします。これにより、コミット情報を、ルート・ファイルではなく .aij ファイルに書き込みます。高速コミット処理では、After-image ジャーナルが必要です。
- スナップショットは、無効にするか、遅延で有効にします。
  - スナップショットを無効にし、ジャーナルの最適化を有効にすると、ルート・ファイルへの書込みはまったく実行しません。
  - スナップショットを遅延、ジャーナルの最適化を有効にすると、アクティブな読取り専用トランザクションでは、読取り専用トランザクションが完了するまでの間、ジャーナルの最適化オプションは無効にします。

ジャーナルの最適化オプションは、更新集中型のデータベース・システムを対象にしたものです。次のような理由から、読取り専用トランザクションが頻繁に発生するシステムでは、ジャーナルの最適化はお薦めしません。

- スナップショットを無効にすると、読取り専用トランザクションによってロックの競合が発生し、トランザクションの処理速度が低下します。
- スナップショットを遅延すると、読取り専用トランザクションでは、ジャーナルの最適化（データベース・ルート・ファイルへの書込みを実行しない）とデフォルト（データベース・ルート・ファイルへの書込みを実行する）の切替えが発生します。この切替えは、重大なオーバーヘッドの原因となります。さらに、読取り / 書込みトランザクションをすべてコミットするまでの間、読取り専用トランザクションはストールします。

ジャーナルの最適化は、データベースの整合性や一貫性を維持することを目的としたトランザクション・シーケンス番号（Transaction Sequence Number：TSN）の割当てと使用など、Oracle Rdb の基本アーキテクチャに適合しています。ジャーナルの最適化オプションを無効にすると、一度に TSN を 1 つユーザーに割り当てます。有効にすると、既定の範囲の TSN を各ユーザーに割り当てます。各ユーザーに既定の範囲の TSN を割り当てることにより、トランザクションごとにコミット情報をデータベース・ルートに書き込む必要がなくなるので、シングル・スレッドの問題が解消されます。すべてのトランザクション情報は、各ユーザーに割り当てられた TSN 範囲を除いて、.aij ファイルに書き込まれます。割り当てられた範囲は、ルート・ファイルに書き込まれます。TSN が不足すると、新しい TSN 範囲が割り当てられます。TSN 範囲のサイズは、ジャーナルの最適化を有効にするときの

TRANSACTION INTERVAL IS n 修飾子で指定します。n は、TSN の数を示します。構文の詳細は、4.1.5.4 項（4-100）を参照してください。

トランザクション間隔の値（TSN の範囲）は、8 ～ 1024 で指定します。デフォルト値は 256 です。トランザクション間隔の指定では、次のような点を考慮してください。

- トランザクション間隔に大きな値を指定すると、小さい値を指定した場合よりも TSN が不足する頻度が減るので、ルート・ファイルを更新（ユーザーが新しい範囲の TSN を割り当てるときに必要）する I/O の数が減ります。逆に、小さな値を指定すると、ルート・ファイルへの I/O が増加します。I/O が増加すると、データベース・パフォーマンスは低下します。
- トランザクション間隔に大きな値を指定すると、プロセスは、割り当てられた TSN を効率的に使用できなくなる可能性があります。トランザクション間隔が 512 で、データベースのアクティブ・ユーザーは 1 日あたり 512 人だとします。このデータベースでは、512 の TSN の中で実際に使用しているものはごく少数であるとしても、1 日あたり 262,144 の TSN を使用することになります。さらに、512 人のユーザーが、データベースのオープンとクローズを 1 日に 4 回実行する場合、1 日あたりの TSN は 1,048,576 になります。したがって、TSN が不足し、TSN の値のリセットが必要になります。
  逆に、トランザクション間隔に小さな値を指定すると、TSN をリセットする間隔が長くなります。

使用しているデータベース環境では、パフォーマンスと TSN の不足のどちらを優先するかを考えてください。一般的なガイドラインとして、データベースのユーザー数が少ないか、すべてのユーザー・セッションが長い場合には、トランザクション間隔に大きな値を指定します。データベースのユーザー数が多いか、すべてのユーザー・セッションが短い場合には、トランザクション間隔に小さな値を指定します。

トランザクション間隔が最適な値かどうかを判断するには、値を 1 つ選択し、ルート・ファイルへの I/O をモニターしてください。次に、トランザクション間隔の値を増減し、モニター結果を前の値と比較して、調整していきます。

4.1.5.4 高速コミット・トランザクション処理の有効化

高速コミット処理やジャーナルの最適化を有効にするには、SQL ALTER DATABASE 文を使います。次は、構文の例です。

```
SQL> ALTER DATABASE FILENAME test1
  1>    JOURNAL FAST COMMIT {ENABLED | DISABLED}
  2>            (CHECKPOINT INTERVAL IS n BLOCKS,
  3>             CHECKPOINT TIMED EVERY s SECONDS
  4>             [NO] COMMIT TO JOURNAL OPTIMIZATION,
  5>             TRANSACTION INTERVAL IS m);
```

n には、次のチェックポイントを実行するまでの間、.aij ファイルに追加できるブロック数を指定します。s には、次のチェックポイントまでの時間を秒数で指定します。m には、

ジャーナルの最適化を有効にした場合に、各ユーザーに事前に割り当てる TSN の数を指定します。

注意：TRANSACTION INTERVAL IS パラメータは、ジャーナルの最適化に条件を加えるものであり、チェックポイント間隔を指定する方法ではありません。また、TRANSACTION INTERVAL IS パラメータと、論理名 RDM$BIND_CKPT_TRANS_INTERVAL や構成パラメータ RDB_BIND_CKPT_TRANS_INTERVAL を混同しないように注意してください。RDM$BIND_CKPT_TRANS_INTERVAL と RDB_BIND_CKPT_TRANS_INTERVAL では、チェックポイント間隔としてトランザクション・カウントを指定します。

チェックポイント・パラメータは、一方または両方の指定ができます。次は、高速コミット処理を有効にし、チェックポイント間隔として 256 ブロック、120 秒、トランザクション・カウントに 20 を指定した例です。また、ジャーナルの最適化を有効にし、各ユーザーに 512 の TSN を割り当てています。

```
$ DEFINE RDM$BIND_CKPT_TRANS_INTERVAL 20
$ SQL
SQL> ALTER DATABASE FILENAME test1
  1>    SNAPSHOT IS DISABLED
  2>    JOURNAL FILENAME DISK2:[USER]test2.aij
  3>    JOURNAL FAST COMMIT ENABLED
  4>          (CHECKPOINT INTERVAL IS 256 BLOCKS,
  5>           CHECKPOINT TIMED EVERY 120 SECONDS
  6>           COMMIT TO JOURNAL OPTIMIZATION,
  7>           TRANSACTION INTERVAL IS 512);
```

高速コミット処理を使用するには、After-image ジャーナルを有効にする必要があります。ジャーナルの最適化を有効にするには、スナップショットを無効に（または遅延）する必要があります。AIJ バッファを有効にしないと、高速コミットやジャーナルの最適化はアクティブになりません。スナップショットを無効または遅延しないと、ジャーナルの最適化はアクティブになりません。

次は、高速コミットは有効ですが、ジャーナルの最適化は無効にした例です。

```
SQL> ALTER DATABASE FILENAME test1
  1>    JOURNAL FAST COMMIT ENABLED
  2>          (CHECKPOINT INTERVAL IS 256 BLOCKS,
  3>           CHECKPOINT TIMED EVERY 120 SECONDS
  4>           NO COMMIT TO JOURNAL OPTIMIZATION);
```

高速コミット処理やジャーナルの最適化を無効にする場合も、後で有効にした場合に使用できるように、チェックポイント間隔や TSN 範囲を指定しておいてください。ジャーナルの最適化オプションを使用するには、高速コミット処理を有効にする必要があります。次の例

では、高速コミットが無効であるため、ジャーナルの最適化は有効になっているようにみえますが、実際には無効です。

```
SQL> ALTER DATABASE FILENAME test1
  1>    JOURNAL FAST COMMIT DISABLED
  2>          (CHECKPOINT INTERVAL IS 256 BLOCKS,
  3>           CHECKPOINT TIMED EVERY 120 SECONDS
  4>           COMMIT TO JOURNAL OPTIMIZATION,
  5>           TRANSACTION INTERVAL IS 512);
```

次の文を入力すると、両方の機能を有効にし、パラメータを設定できます。

```
SQL> ALTER DATABASE FILENAME test1
  1>    JOURNAL FAST COMMIT IS ENABLED
  2>      (COMMIT TO JOURNAL OPTIMIZATION);
```

高速コミット処理の詳細な構文は、『Oracle Rdb7 SQL Reference Manual』を参照してください。

高速コミットの設定は、Performance Monitor の「Fast Commit Information」画面で確認できます。次は、「Fast Commit Information」画面の例です。

```
Node: TRIXIE         Oracle Rdb V7.0-00 Performance Monitor 24-JUN-1996 11:43:21
Rate: 3.00 Seconds          Fast Commit Information         Elapsed: 00:00:32.56
Page: 1 of 1       RDBVMS_USER1:[LOGAN.V70]MF_PERSONNEL.RDB;1       Mode: Online
--------------------------------------------------------------------------------
AIJ Fast Commit is enabled
- Checkpointing AIJ interval is 256 blocks
- Checkpointing time interval is 120 seconds
Commit to AIJ optimization is enabled
- Transaction interval is 512
--------------------------------------------------------------------------------
Exit Help Menu Options Refresh Set_rate Write !
```

「Fast Commit Information」画面は、「Database Parameter Information」サブメニューから選択できます。

高速コミット処理の設定は、SHOW TRANSACTION 文でも確認できます。高速コミット処理を有効にした状態で、SET TRANSACTION 文を使って、共有読取り用の EMPLOYEES テーブルを予約する読取り / 書込みトランザクションを開始すると、SHOW TRANSACTION 文は、次のような内容を表示します。

```
SQL> SHOW TRANSACTION
Transaction information:
    Statement constraint evaluation is off
On the default alias
Transaction characteristics:
        Read Write
        Reserving table EMPLOYEES for shared read
```

```
Transaction information returned by base system:
a read-write transaction is in progress
  - updates have not been performed
  - fast commit is enabled
  - aij commit is enabled
  - transaction sequence number (TSN) is 137
  - snapshot space for TSNs less than 137 can be reclaimed
  - session ID number is 82
SQL>
```

チェックポイントに関する統計は、Performance Monitor の「Checkpoint Statistics」画面で確認できます（4.1.1.11 項（4-16）を参照してください）。

4.1.5.5 メモリー・ページ転送

メモリー・ページ転送機能を有効にすると、変更したページに他のプロセスがアクセスするまでの間、そのページをディスクに書き込む必要がなくなります。つまり、プロセスの割当てセット内のページには、ディスクへの書込みが完了していない状態ならば、他のプロセスがコミットした更新内容を格納できるということです。データベースが、メモリー・ページ転送で必要な条件を満たしている場合、更新集中型のアプリケーションでは、パフォーマンスが大幅に向上する可能性があります。これは、更新内容のディスクへの書込みを省略し、プロセス間でページを共有することによって、入出力（I/O）の回数を減らすことができるからです。

メモリー・ページ転送機能は、次のような特徴を持ったデータベースで使用できます。

- グローバル・バッファを有効化
- After-image ジャーナルを有効化
- 高速コミットを有効化
- データベースへのアクセスは、シングル・ノードのみ（NUMBER OF CLUSTER NODES の値は 1）

データベースが上記の条件を満たしている場合、SQL CREATE DATABASE、ALTER DATABASE、IMPORT 文の PAGE TRANSFER VIA MEMORY 句を指定すると、メモリー・ページ転送を有効化できます。上記の条件を満たしていないデータベースで PAGE TRANSFER VIA MEMORY 句を指定しても、メモリー・ページ転送は有効化されません。

Oracle Rdb バージョン 6.1 以下では、プロセス間のページ転送は、必ずディスクを経由していました。PAGE TRANSFER 句のデフォルトは、PAGE TRANSFER VIA DISK であり、バージョン 6.1 以下でこのオプションを使用していたアプリケーションは、6.1 より新しいバージョンでも引き続き使用できます。

Oracle Rdb は、高速コミット処理を有効にすると、チェックポイント・レコードを After-image ジャーナル（.aij）ファイルに書き込みます。トランザクションに障害が発生すると、Oracle Rdb は、現在の障害が発生したトランザクションをロールバックし、.aij ファ

イルに格納されているレコードのうち、最後のチェックポイント・レコード以降にコミットされたトランザクションをすべて再処理します。メモリー・ページ転送機能を有効にすると、Oracle Rdb はチェックポイント・レコードを .aij ファイルに書き込まないので、異なる方法でリカバリ処理を行います。メモリー・ページ転送では、内部データ構造を使用し、各プロセスのチェックポイント仮想ブロック番号（Virtual Block Number：VBN）を追跡します。Oracle Rdb は、データベースのリカバリを実行する際、.aij ファイルの中で、チェックポイント VBN の値が最も小さいものから再処理していきます。

PAGE TRANSFER 句の詳細は、『Oracle Rdb7 SQL Reference Manual』の ALTER DATABASE 文、CREATE DATABASE 文、IMPORT 文を参照してください。

4.1.6 行（レコード）の断片化

レコードの変更によって物理的な行長が長くなった場合、ページに空き領域が不足して、新しい行を格納できなくなると、行は断片化します。特に、行のテキスト・フィールドを頻繁に変更する場合、断片化が発生しやすくなります。たとえば、あるデータベースでは、1 行に多数の空白列があり、行を圧縮して、圧縮した行長を基準にページ・サイズを決定するとします。圧縮した行をページに格納しますが、空の列に値を入力すると、行はページに収まらなくなります。Oracle Rdb は、行を断片化し、テーブルの論理領域の中で使用可能なページを探し、そこに格納します。行の断片にはポインタがあり、他の断片の位置を示します。このような構造によって、Oracle Rdb は、データベース・ページを効率的に使用しています。ただし、断片化した行を取り出す場合、すべての断片を探して 1 つの完全な行に組み立てる必要があるため、1 行全体を取り出すよりも I/O 時間は長くかかります。

したがって、行が断片化しないように注意してください。たとえば、データベースの各記憶領域について、通常の圧縮していない状態で、最長の行を探します。ある記憶領域の行が 1800 バイトだとすると、この記憶領域のページ・サイズには、データとオーバーヘッドを格納できるだけの大きさを設定する必要があるので、4 ブロックとなります。

RMU Analyze コマンドを使用すると、データベースの行が断片化されているかどうかを確認できます。行の断片化は、次のような状態で行が変更されるときに発生します。

- 行の格納
  新しく格納した行が大きすぎると、1 つのデータベース・ページに入りません。未使用のデータベース・ページにある空き領域が十分でないと、行全体を格納できなくなり、行は断片化します。記憶領域 PAGE SIZE に対して前の行サイズを基準に指定した値が小さすぎる場合は、SQL CREATE STORAGE AREA 文で新しい記憶領域を作成し、PAGE SIZE パラメータで新しいページ・サイズを指定し、SQL ALTER STORAGE MAP 文ですべての行を新しい記憶領域にマッピングします。SQL EXPORT と IMPORT 文を使うと、すべての記憶領域をアンロードして再ロードするので、こちらの方法をお薦めします。
  新しい行を作成するとき、その行の数列にしかデータ値を格納しなければ、すべての行に値を格納した場合よりも、データベース・ページ上で必要になる領域は少なくなります。作成後、その行を読み取って、空白行に実際の値を格納すると、データベース・

ページ上で必要な領域は増加します。したがって、行に含まれる列には、できるだけデータ値を格納するようにしてください。

- 行の変更
  データベースでは、行のデータ型や行サイズを変更でき、行には列を追加できるので、新しく作成した行のサイズが、既存のどの行よりも大きくなる可能性があります。Oracle Rdb は、このようにサイズが大きくなった新しい行をデータベース・ページに書き戻すとき、空き領域が十分にない場合は、2 つのデータベース・ページに行を分割することがあります。

Oracle Rdb は、データベースから行を削除する場合、その行に対応し、ページ上にあるライン・インデックス・エントリを変更します。ライン・インデックス・エントリは、空き状態としてマークされますが、行を削除したトランザクションのために予約されます。トランザクションが COMMIT 文で終了すると、空き領域は永続的になります。dbkey スコープの指定がなく、デフォルトの SQL DBKEY SCOPE IS TRANSACTION オプションを使用する場合、他のトランザクションはこの空き領域を使用できます。トランザクションをロールバックする際は、この空き領域に行を挿入します。

4.1.7 リカバリ・バッファの数の指定

リカバリ・バッファの数のデフォルト値は 20 です。データベース・アプリケーションでは、個々のアクティブ・ユーザーに Recovery-unit ジャーナルを実行し、ユーザーによる制御はできません。したがって、データベース・アプリケーションのアクティブ・ユーザーと同じ数の .ruj ファイルが作成されます。ユーザーのトランザクションをロールバックする際は、そのユーザーの .ruj ファイルを使って、プロセスをロールバックします。これよりも規模は大きくなりますが、システム障害が発生してデータベース・アクセスが中断された場合、Oracle Rdb は、システムが復旧した後、すべてのユーザーの .ruj ファイルを使ってロールバック（リカバリ）し、障害時にコミットされていなかった更新トランザクションを再処理します。

このようなデータベース・リカバリ・プロセスを実行するタイミングは、アクティブ・ユーザーの数（.ruj ファイルの数）、トランザクションの長さ（.ruj ファイルのサイズ）、リカバリ・バッファの数（リカバリ専用メモリー）によって決まります。更新トランザクションの長さが比較的短い場合は、リカバリ・プロセスにあまり時間はかかりません。更新トランザクションが比較的長い場合は、リカバリ・プロセスに時間がかかることもあります。リカバリ・バッファの数をできるだけ大きい値に設定すると、1 回の I/O 操作でバッファにロードできる情報量が多くなるので、リカバリ・プロセス全体の時間を大幅に短縮できます。

OpenVMS VAX OpenVMS Alpha

リカバリ・バッファには、DBR プロセスのワーキング・セットと同等のサイズを選択してください。SQL ALTER DATABASE 文を使って、バッファ・カウントを DBR プロセスの WSEXTENT とほぼ同じ値に調整します。つまり、DBR プロセスの WSQUOTA を十分大きな値に設定すれば、データベース・リカバリが必要になったときに、DBR プロセスは、指定された物理ページ数を使用できるようになります。DBR プロセスの WSEXTENT をできるだけ大きな値に設定し、指定された物理ページの最大数を使用できるようにしてください。

VMScluster 環境でのデータベース・リカバリの詳細は、6 章（6-1）を参照してください。ワーキング・セット・パラメータの設定や、ワーキング・セット・クォータ・パラメータの自動チューニングに関するガイドラインは、OpenVMS のドキュメントを参照してください。◆

NUMBER OF RECOVERY BUFFERS の変更については、『Oracle Rdb7 Guide to Database Maintenance』を参照してください。

4.1.8 After-image ジャーナル・ファイルの割当て

.aij ファイルは、SQL の CREATE DATABASE 文で最初にデータベースを作成する時点では、デフォルトで無効化されています。SQL の ALTER DATABASE 文を使用し、JOURNAL IS オプションでファイル名を指定してください。.aij ファイルのデフォルトの割当て領域は、0 ブロックです。.aij ファイルでは、次の 3 点に注意してください。

- 領域の割当て。
- 領域の割当ての拡張。
- .aij ファイルをバックアップすることにより、AIJ ディスクがいっぱいになることを防ぐ。

.aij ファイルの作成では、割当てとエクステントを指定できます。割当ては、ディスク上に割当てられる物理ファイルのサイズを定義します。

エクステント・パラメータは、ファイルの物理的な拡張が必要になった場合の物理サイズを定義するだけでなく、割り当てられた物理ファイル内での論理的な end-of-file（EOF）の定義にも使用します。たとえば、.aij ファイルを作成し、割当てには 10,000 ブロック、エクステントには 1000 ブロックを定義したとします。ファイルは、1000 ブロックごとに管理されます。つまり、ファイルが実際に物理的に拡張されるまでには、論理的な拡張および初期化を 10 回行うことになります。ファイルが物理的に拡張されていなくても、論理的に拡張されていれば、Performance Monitor で表示されます。ファイルは、論理的な EOF まで初期化されます。

AIJ ディスクがいっぱいになるのを防ぐには、ファイルが物理的に拡張される前に、RMU Backup After_Journal コマンドを実行してください。つまり、AIJ アクティビティが大量に発生する環境では、Threshold 修飾子に、ファイルの拡張が不要で、しかもディスクに収まる程度の値を選びます。たとえば、Threshold=7000 修飾子を指定して RMU Backup After_Journal コマンドを実行すると、AIJ バックアップのしきい値は 7000 ブロックになります。バックアップ・プロセスは、catch up モードで .aij ファイルのバックアップを起動するので、.aij ファイルの EOF が Threshold 値よりも小さければ、他のトランザクションをブロックしません。.aij ファイルの EOF が Threshold 値を超える場合にのみ、静止ポイント・ロックが要求されます。静止ポイント・ロックが割り当てられると、.aij ファイルのバックアップが完了するまで、トランザクションはブロックされます。.aij ファイルのバックアップと切捨てを行い、論理的な EOF は 0 にリセットされます。論理的な EOF が物理的な EOF に達することはないので、.aij ファイルは物理的に拡張されることはなく、ディスクもいっ

ぱいになりません。.aij ファイルのバックアップの詳細は、『Oracle Rdb7 Guide to Database Maintenance』を参照してください。

バックアップ・プロセスは、Threshold 値に達する前に .aij ファイルのバックアップを起動し、他のトランザクションをブロックしません。EOF が Threshold 値を超えたために静止ポイント・ロックが要求され、割り当てられるときにだけ、アクティブなトランザクションがブロックされます。したがって、1 つのトランザクションが完了する前に、.aij ファイルがいっぱいになって物理的に拡張される可能性もあります。このような場合には、エクステント・パラメータが役立ちます。エクステント・パラメータが大きすぎると、物理的に割り当てられる領域も大きくなります。エクステント・パラメータが小さすぎると、頻繁に拡張が発生し、パフォーマンスが低下します。

トランザクション件数が多いアプリケーションでは、割当てサイズを大きくすると、拡張の頻度を減少させることができるので、パフォーマンスは大幅に向上します。更新が大量に発生し、応答時間を重視するアプリケーションでは、.aij ファイルの事前割当てを行い、通常の処理では拡張が起こらないようにしてください。1 日分の処理に対応できるだけのサイズを設定し、RMU Backup After_Journal コマンドで、1 日 1 回 .aij ファイルをテープにアンロードします。.aij ファイル専用のディスク・デバイスがある場合は、ディスク全体の容量を上限として、割当てサイズを設定してください。

4.1.9 スナップショット・ファイルの割当て

スナップショット・ファイルの割当てサイズは、データベース全体のサイズは SQL の CREATE DATABASE 文、個々の記憶領域でのサイズは SQL の CREATE STORAGE AREA 文で設定できます。

最初に、スナップショットの割当てのデフォルト値は 100 ページなので、これをデータベース全体のサイズとして割り当てた場合、全記憶領域の読取り専用ユーザーに十分対応でき、しかも拡張を行わなくてよいかどうかを確認します。または、各記憶領域について、読取り専用ユーザーのアクセス頻度と読取り / 書込みユーザーのアクセス頻度を比較し、デフォルトの 100 ページを調整する方法もあります。更新トランザクションは、更新する行の Before-image をスナップショット・ファイルに書き込みます。したがって、記憶領域にあるテーブルで実行するトランザクション速度と更新処理がわかれば、スナップショット・ファイルが増大する速さとサイズを推測できます。このように、記憶領域にスナップショット・ファイルの割当てを設定してください。

スナップショット・ファイルが大きくなりすぎる場合、RDB$SYSTEM 記憶領域のスナップショット・ファイルに SQL の ALTER DATABASE 文の SNAPSHOT ALLOCATION IS オプションを使用し、割当てサイズを小さい値に設定すれば、ファイル・サイズを小さくできます。また、SQL の ALTER DATABASE ALTER STORAGE AREA 文の SNAPSHOT ALLOCATION IS オプションを使用すると、他の記憶領域のスナップショットの割当てサイズを変更できます。また、スナップショット・ファイルの拡張を遅延する方法もあります。これには、ALTER DATABASE 文の SNAPSHOT IS ENABLED DEFERRED オプションを使

用します。このように、読取り / 書込みトランザクションは、読取り専用トランザクションの実行中、Before-image のみをスナップショット・ファイルに書き込みます。

4.1.10 After-image ジャーナル・ファイルとスナップショット・ファイルのエクステント

パフォーマンスを低下させないために、.aij ファイルと .snp ファイルには、最初から十分なサイズを割り当ててください。ただし、拡張が発生した場合には、それ以上拡張が発生しないようにエクステント・オプションを設定し、パフォーマンスがさらに低下しないようにします。エクステント・オプションに十分なサイズを指定し、ファイルを 1 回拡張するだけで対応できるようにしてください。

4.1.11 スナップショット・ファイルへのアクセス

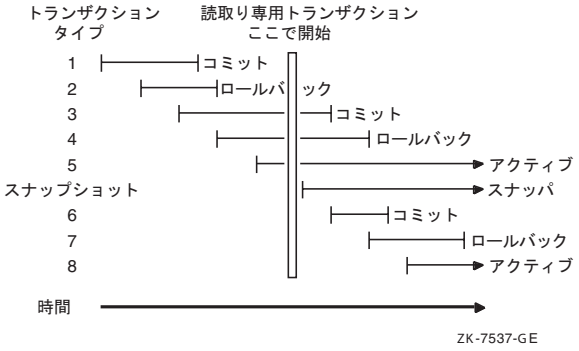
SQL の SET TRANSACTION 文で READ ONLY を指定すると、データベースのスナップショット・ファイルにアクセスできます。読取り専用トランザクションは、トランザクションの開始時点での一貫したビュー（ある時点でのスナップショット）を提供します。変更内容の Before-image を .snp ファイルに書き込むのは、共有または保護付きの書込みトランザクション（排他的書込みではない）のみなので、読取り専用トランザクションは、イメージをすぐに参照でき、このような書込みトランザクションがロックを解放するまで待つ必要はありません。また、読取り専用トランザクションは、一貫したビューを維持するために、データの読取りロックを保持する必要がありません。このように、読取り専用トランザクションには、次のような利点があります。

- ある時点でのデータの一貫したビュー
- ロックの競合を最小限にする

読取り専用トランザクションに正しいバージョンの行（読取り専用トランザクションが起動する前にコミットされているバージョン）を提供する方法として、Oracle Rdb は TSN を使用します。Oracle Rdb は、起動した個々のトランザクションに TSN を割り当てます。トランザクションがデータベースを更新すると、そのトランザクションの TSN は、新しいバージョンの行に付加されます。

図 4-10（4-109）は、読取り専用トランザクションのスケジュールと、トランザクションの各タイプについて、読取りトランザクションがどのように認識するかを示しています。

図 4-10 データベース・スナップショット・ファイルにアクセスするトランザクション



スナップショット・ファイルには、読取り専用トランザクションが開始した時点ですでにコミットされているトランザクションの更新のみが格納されています。これには、タイプ 1 のトランザクションが相当します。

更新がスナップショット・ファイルに格納されていないトランザクションは、次のようなタイプに分類されます。

- 読取り専用トランザクションが開始した時点で、すでにアクティブだったトランザクション。これには、タイプ 3、4 および 5 のトランザクションが相当します。
- 読取り専用トランザクションが開始した後に、起動したトランザクション。これには、タイプ 6、7 および 8 のトランザクションが相当します。
- ロールバックしたトランザクション。これには、タイプ 2 のトランザクションが相当します。

4.1.12 Oracle Rdb でのスナップショット・ファイル・オプション

大規模なデータベース更新プログラムの実行やマルチユーザーによる更新が大量に発生すると、トランザクションを完了するために必要な I/O 操作が大量に発生し、全体的なデータベース・パフォーマンスに影響を与えます。Oracle Rdb では、データベース行の変更を大量に実行するロード・プログラムや更新プログラムは、特定のディスク・デバイスの I/O 操作機能をフルに活用できます。スナップショット・ファイルへの書込みを無効にすることにより、すべての I/O 操作は、データベース本体へのアクセスに集中します。

注意: スナップショット・ファイル（.snp）は絶対に削除しないでください。削除すると、データベースのバックアップやリストア操作を実行したデータベースが破損します。.snp ファイルを削除すると、データベース・ルート・ファイル（.rdb）に格納されている .snp ファイルへのポインタは更新されません。スナップショット・ファイルを使用しない場合は、SQL の ALTER DATABASE 文でスナップショットを無効にしてください。

データベースの更新アクセスが同時に大量に発生すると、Oracle Rdb は、データベース、.snp ファイル、.ruj ファイル、After-image ジャーナルを有効にしている場合は .aij ファイルに書込みを実行しなくてはなりません。スナップショットを無効にすると、スナップショット・ファイルへの書込みを省略でき、1 つのトランザクションで必要な I/O 操作の数を減らすことができます。更新時の排他オプションを使用すると、スナップショットを一時的に無効にできます。スナップショットの無効化は、更新集中型のアプリケーションのパフォーマンスを向上する最後の手段として使用してください。他の方法として、.snp ファイルを記憶領域ファイル（.rda）とは別のディスク上に格納することもできます。ほとんどの場合、ロックやデッドロック状態を回避できるというスナップショットの利点は、スナップショット・ファイルの I/O オーバーヘッドよりも重要です。

次のように、SNAPSHOT IS DISABLED オプションを指定すると、.snp ファイルへのユーザー・アクセスや書込み操作を有効または無効にできます。

- SQL CREATE DATABASE
- SQL ALTER DATABASE
- SQL IMPORT

スナップショット・ファイルを有効にすると、読取り / 書込みトランザクションは、更新したレコードの Before-image をスナップショット・ファイルに書き込みます。読取り / 書込みトランザクションの後に読取り専用トランザクションが起動し、読取り / 書込みトランザクションが変更したレコードを読み取ろうとすると、Oracle Rdb は、スナップショット・ファイルに格納された Before-image を提供します。読取り専用トランザクションは、ロックを使わないでレコードを読み取ります。

スナップショット・ファイルを無効にすると、読取り専用トランザクションと読取り / 書込みトランザクションは、有効にした場合とは異なる手順で処理を行います。スナップショット・ファイルを無効にすると、読取り / 書込みトランザクションは、更新したレコードの Before-image をスナップショット・ファイルに書き込みません。したがって、読取り専用トランザクションは、更新レコードの Before-image を参照できなくなります。読取り専用トランザクションは、読取り / 書込みトランザクションと同様に、実際のデータ・レコードを読み取る必要があるので、データ・レコードの一貫したビューを取得するために、読取り専用トランザクションもロックを使用します（読取り / 書込みトランザクションと同様）。スナップショット・ファイルを無効にした場合の読取り専用トランザクションは、特殊なタイプの読取り / 書込みトランザクションだと考えられます。読取り / 書込みトランザクション

と同様にデータ・レコードをロックしますが、データベースの更新はできません。スナップショット・ファイルを無効にした場合の読取り専用トランザクションは、データ・レコードのロックを共有するので、他のユーザーも読み取り中のレコードにアクセスできます。

スナップショット・ファイルを無効にすると、読取り / 書込みトランザクションはデータ・レコードを更新でき、更新がコミットされるかロールバックされるまでの間は、他のトランザクションがこのデータ・レコードにアクセスしようとすると、ロックの競合が発生します。また、他のテーブルから行を取り出そうとして、更新トランザクションが索引ノードをロックしている場合は、一部のデータベースにアクセスできない可能性もあります。

他の更新トランザクションがテーブル内の行を順番に更新していく場合、Oracle Rdb は、そのトランザクションが完了するまで、テーブル全体をロックします。したがって、スナップショットを無効にした場合、有効にした場合とは、読取り専用の取出し操作は異なる方法で実行されます。

これまでに説明したように、多数のユーザーが行の取出しを行うマルチユーザー・アクセスが重要でない場合には、データベース更新操作のパフォーマンスを向上するために、スナップショットを無効化できます。バッチ処理、トランザクション処理、更新プログラムなど、対話型アクセスがあまり発生しない処理が対象になります。このような更新処理が完了し、効率的な同時アクセスが再度要求された時点で、スナップショットを有効にできます。データベース更新操作の間、マルチユーザー・アクセスが必要なければ、排他的トランザクションを使用するとよいでしょう。排他的トランザクションを使用すると、更新操作でのパフォーマンスを最適化でき、SQL の ALTER DATABASE 文や IMPORT 文でスナップショットを無効にする必要はありません。

例 4-14（4-111）では、SQL の CREATE DATABASE 文を使用して、データベース・ファイル DR2:[RDB]mf_personnel.rdb を作成し、定義した記憶領域 RDB$SYSTEM に、.snp ファイルである DR2:[RDB]mf_pers_default.snp をデフォルトで作成および有効にしています。

例 4-14 RDB$SYSTEM 記憶領域のスナップショット・ファイルの作成

```
SQL> CREATE DATABASE FILENAME 'DR2:[RDB]mf_personnel'
  1> CREATE STORAGE AREA RDB$SYSTEM FILENAME 'DR2:[RDB]mf_pers_default.rda';
```

例 4-15（4-111）の SQL の CREATE DATABASE 文は、明示的な修飾子を使用して、.snp ファイルを作成および有効にしています。

例 4-15 RDB$SYSTEM 記憶領域のスナップショット・ファイルの作成と明示的な有効化

```
SQL> CREATE DATABASE FILENAME 'DR2:[RDB]mf_personnel' SNAPSHOT IS ENABLED;
```

十分な領域がなくてすべてのデータベース・ファイルを格納できない場合や、.snp ファイルが必要でない場合には、.snp ファイルの初期割当てサイズに小さな値、または 0 ページを指定できます。次のような場合には、スナップショット割当てを 0 に設定すると便利です。

- スナップショットを無効にし、使用可能な領域を増やしたい
- 読取り / 書込み用の記憶領域を読取り専用に変更し、スナップショット・ファイルを使用しないので、使用可能な領域を増やしたい

例 4-16（4-112）のように、SQL の CREATE DATABASE 文の最初のオプションを使用すると、.snp ファイルで使用するディスク容量を制御できます。

例 4-16 RDB$SYSTEM 記憶領域のスナップショットの作成 --- サイズの上限を 50 ページに指定

```
SQL> CREATE DATABASE FILENAME 'DR2:[RDB]mf_personnel'
  1>   SNAPSHOT IS ENABLED
  2>   CREATE STORAGE AREA RDB$SYSTEM FILENAME 'DR2:[RDB]mf_pers_default.rda'
  3>      SNAPSHOT ALLOCATION IS 50 PAGES
  4>      SNAPSHOT EXTENT IS 200 PAGES;
```

上記の SQL の CREATE DATABASE 文と CREATE STORAGE AREA 文は、次のような処理を行います。

- データベース・ファイル DR2:[RDB]mf_personnel.rdb の作成
- .snp ファイルへの書込みを有効化
- .rda ファイル DR2:[RDB]mf_pers_default.rda の作成
- mf_pers_default.rda 記憶領域に、.snp ファイル DR2:[RDB]mf_pers_default.snp を作成
- .snp ファイルの割当てを 50 ブロックに設定
- .snp ファイルのエクステント・サイズを 200 ページに設定

例 4-17（4-112）は、SQL の ALTER DATABASE 文で、2 つのデータベース全体のパラメータを変更し、次のような処理を実行します。

- .snp ファイルへの書込みを無効化
- .aij ファイル名を指定して、After-image ジャーナルを有効化

例 4-17（4-112）では、SQL の ALTER DATABASE 文を使用して、.snp ファイルの割当てとエクステント・サイズを変更しています。

例 4-17 スナップショット・ファイルの割当てとエクステント・サイズの変更

```
SQL> ALTER DATABASE FILENAME mf_personnel.rdb
  1>    SNAPSHOT IS DISABLED
  2>    JOURNAL FILE DR3:[RDB]newpers.aij
  3>    SNAPSHOT ALLOCATION IS 3 PAGES
  4>    SNAPSHOT EXTENT IS 200 PAGES;
```

この SQL の ALTER DATABASE 文は、次のような処理を実行します。

- スナップショットを無効化
- 新しい .aij ファイルである DR3:[RDB]newpers.aij を作成して有効化

- .snp ファイルの割当てに小さい値を設定し、.snp ファイルを切り捨てる
  RDB$SYSTEM 記憶領域に現在設定されている .snp ファイルの割当てサイズが必要以上に大きい場合、Oracle Rdb はファイルを切り捨てます。したがって、.snp ファイルの割当てサイズに小さな値を指定し、ディスク領域を再要求できます。
- スナップショット・ファイルのエクステント・サイズを指定

例 4-18（4-113）では、SQL の ALTER DATABASE 文を使用して、マルチボリューム・データベースで、スナップショットのエクステント・サイズを指定しています。

例 4-18 マルチボリューム・データベースでのスナップショット・エクステント・サイズの指定

```
SQL> ALTER DATABASE FILENAME mf_personnel.rdb
  1>    SNAPSHOT IS ENABLED
  2>    JOURNAL FILE DR3:[RDB]newpers.aij
  3>    SNAPSHOT ALLOCATION IS 200 PAGES
  4>    SNAPSHOT EXTENT IS (MINIMUM OF 200 PAGES,
  5>    MAXIMUM OF 1000 PAGES, PERCENT GROWTH IS 15);
```

4.1.13 遅延スナップショットの機能

一般的に、読取り専用トランザクションや読取り / 書込みトランザクションが同時に発生するデータベースでは、スナップショットを有効にするとパフォーマンスが向上します。読取り専用ユーザーは、行をロックしなくてもデータを取り出せるので、他のユーザーもデータへアクセスでき、一貫したビューを参照できます。つまり、読取り専用ユーザーは、読取り専用トランザクションが起動する前にコミットされているトランザクションが更新した行を参照することはできますが、読取り専用トランザクションがアクティブである間、行への変更内容は参照できません。一般的に、同時ユーザー環境では、読取り専用ユーザーがレコード・ロックの競合に遭遇したり、競合を発生させることがありますが、このような環境での競合を少なくすることにより、パフォーマンスは向上します。

ただし、スナップショットを有効にすると、共有または保護付きの読取り / 書込みトランザクションは、更新する行のコピーを 1 つ、.snp ファイルに書き込まなくてはなりません。このように、読取り / 書込みトランザクション 1 件あたり、I/O 操作 1 つ分のオーバーヘッドが増えますが、読取り専用ユーザーがデータベースにアクセスしている場合には効果的です。読取り専用のユーザー・アクセスがない場合（または、読取り専用ユーザーによるアクセスがほとんどなく、トランザクション・サイズも短い場合）、遅延スナップショットを有効にすることにより、スナップショット・ファイルへの I/O 操作を軽減できます。遅延スナップショットを有効にすると、読取り専用のトランザクションが開始するまで、スナップショット・ファイルへの書込みを遅延します。

注意：読取り専用トランザクションは、読取り / 書込みトランザクションとの間でロックが発生しないということが、パフォーマンス上の利点になっています（SQL READ WRITE ...RESERVING < テーブル名 > EXCLUSIVE トランザクションを除く）。つまり、ロックがパフォーマンス上の問題になっている場合は、スナップショットを有効にすると効果的です。ボトルネックがディスク I/O 操作であれば、スナップショットを有効にすると、パフォーマンスが低下する可能性があります。ディスク I/O のボトルネックは、.snp ファイルを別のディスクに移動することで解消できます。ディスク I/O を軽減するには、グローバル・バッファを有効にするのも効果的です。このような方法でもパフォーマンスが向上しない場合は、遅延スナップショットで問題を解消できる可能性があります。ディスク I/O のボトルネックを解消できず、ロックの問題もない場合は、スナップショットを無効にすると解消できることがあります。スナップショット・ファイルとパフォーマンスの詳細は、8.1.3.6 項（8-37）を参照してください。

遅延スナップショット機能は、SQL の CREATE DATABASE 文、ALTER DATABASE 文、IMPORT 文の SNAPSHOT ENABLED DEFERRED オプションでサポートされます。構文図や参照情報は、『Oracle Rdb7 SQL Reference Manual』または SQL オンライン・ヘルプ・ファイルを参照してください。

デフォルトは、SNAPSHOT ENABLED IMMEDIATE です。スナップショットを即時で有効にすると、読取り / 書込みトランザクションは、更新する行の Before-image を .snp ファイルに書き込みます。読取り専用トランザクションは（予約オプションを指定する場合としない場合）、最初に、データ・ファイルの行を読み取ろうとします。しかし、読取り専用ユーザーがアクセスしたい行が更新対象としてマークされている場合、.snp ファイルの Before-image を読み取ります。読取り / 書込みユーザーのトランザクションを終了するとき、次の 2 つの処理が実行されます。

- 読取り / 書込みユーザーが COMMIT 文を発行すると、更新された行はデータ・ファイルに書き込まれます。ただし、アクティブな読取り専用ユーザーが参照できるのは、.snp ファイルに格納されている行であり、更新内容は参照できません。
- 読取り / 書込みユーザーが ROLLBACK 文を発行すると、読取り / 書込みユーザーの .ruj ファイルから、Before-image である行のコピーが取り出されます。データの変更はないので、アクティブな読取り専用トランザクションは、データ・ファイルに格納されている行の内容を読み取ることができます。

スナップショットを遅延した場合、読取り / 書込みトランザクションがスナップショット・ファイルへの書込みを実行しなくてはならないのは、そのトランザクションが開始した時点で、読取り専用トランザクションがデータベースに接続している場合のみです。読取り / 書込みトランザクションがアクティブである間に読取り専用トランザクションが開始すると、読取り / 書込みトランザクションがコミットまたはロールバックするまでの間、読取り専用

トランザクションは待機しなくてはなりません。アクティブな読取り / 書込みトランザクションは、.snp ファイルへの書込みを行っていないので、読取り専用トランザクションは強制的に待機します。読取り専用トランザクションの待機中に、別の読取り / 書込みトランザクションが起動すると、後から起動した読取り / 書込みトランザクションが、Before-image を .snp ファイルに書き込みます。

表 4-7（4-115）は、SNAPSHOTS DEFERRED を使用する 1 例です。

表 4-7　遅延スナップショットでのトランザクション処理

トランザクション	スナップショットへの書込み	コメント
ユーザーなし		
トランザクション A が、読取り / 書込みを開始	なし	
トランザクション B が、読取り / 書込みを開始	なし	
トランザクション C が、読取り専用を開始		待機
トランザクション D が、読取り / 書込みを開始	あり	
トランザクション A がコミット		
トランザクション B がコミット		
トランザクション C を続行		
トランザクション E が、読取り / 書込みを開始	あり	
トランザクション C がコミット		
トランザクション F が、読取り / 書込みを開始	なし	

スナップショット・モードには、次の 3 種類があります。

- SNAPSHOT DISABLED
  トランザクションは、.snp ファイルへの書込みを実行しません。読取り専用トランザクションは、読取り / 書込みに変換されます。これにより、更新集中型の環境では、I/O オーバーヘッドが軽減されます。ただし、読取り専用トランザクションを変換するので、読取り / 書込みトランザクションと同じようにロックを生成することになり、並列性が低下し、ロックの競合が発生しやすくなります。
- SNAPSHOT ENABLED IMMEDIATE
  共有および保護付きの読取り / 書込みトランザクションは、読取り専用トランザクションのアクセスに関係なく、Before-image をスナップショット・ファイルに書き込みます。.snp ファイルには、更新された行の古いバージョンが格納されているので、読取り専用トランザクションがロックされることはありません。したがって、並列性は高くなりますが、I/O も増加します。
  排他的トランザクションとバッチ更新トランザクションは、性質上、同時に実行できな

いので、.snp ファイルへの書込みを行いません。詳細は、3.8.3 項（3-62）と 3.8.3.5 項（3-74）を参照してください。

- SNAPSHOT ENABLED DEFERRED
  アクティブなユーザーは読取り / 書込みユーザーのみである場合、SNAPSHOT ENABLED DEFERRED は、SNAPSHOT DISABLED と同じように機能し .snp ファイルへの書込みは行いません。すでに開始している読取り / 書込みトランザクションがすべて完了するまでの間、読取り専用トランザクションはストールします（読取り / 書込みトランザクションは、スナップショット・ファイルへの書込みを行っていないので、読取り専用トランザクションは待機します）。読取り専用トランザクションが起動したら、データベースは SNAPSHOT ENABLED IMMEDIATE モードと同様に処理されます。読取り専用トランザクションがすべて完了すると、アクティブな読取り / 書込みトランザクションは .snp ファイルへの書込みを続けますが、新しい読取り / 書込みトランザクションは書込みを実行しません。
  SNAPSHOT ENABLED DEFERRED モードは、スナップショットを無効にした状態と即時で有効にした状態を組み合せたモードです。特に、更新集中型トランザクションと読取り集中型トランザクションが混在する環境では、このモードは効果的です。たとえば、あるデータベースでは、日中は更新が大量に発生するとします（読取りはほとんどありません）。オフの時間帯には、バッチ・アプリケーションが、読取り、更新、レポート作成などを実行します。

スナップショットの現在の設定内容は、Performance Monitor の「General Information」画面で参照できます。例 4-19（4-116）は、「General Information」画面の例です。SQL SHOW DATABASE 文を使用しても、スナップショット情報を表示できます。

例 4-19 スナップショット・ファイルの現在の設定の確認

```
Node: TRIXIE          Oracle Rdb V7.0-00 Performance Monitor 24-JUN-1996 13:20:16
Rate: 3.00 Seconds             General Information           Elapsed: 00:00:13.03
Page: 1 of 1        RDBVMS_USER1:[LOGAN.V70]MF_PERSONNEL.RDB;1       Mode: Online
------------------------------------------------------------------------------
Database created at 30-MAY-1996 14:20:42.77
Maximum user count is 50
Maximum node count is 16
Database open mode is Automatic
Database close mode is Automatic
Snapshot mode is Deferred
Statistics collection is enabled
Active storage area count is 10
Reserved storage area count is 0
Default recovery-unit journal filename is "Not Specified"
Date of last backup is 17-NOV-1858 00:00:00.00
Fast incremental backup is enabled
------------------------------------------------------------------------------
Exit Help Menu Options Refresh Set_rate Write !
```

「General Information」画面は、「Database Parameter Information」サブメニューから選択できます。

4.2 記憶領域パラメータの調整

この項では、記憶領域パラメータと、現在のデータベース・アプリケーション環境に最適な値を選択する方法を説明します。Oracle Rdb は、記憶領域パラメータをデフォルト値に設定しても十分なパフォーマンスを発揮できるので、特殊なアプリケーション向けにデータベースを調整するときのガイドラインとして活用してください。たとえば、アプリケーションが、セグメント化された文字列データ・タイプを多用する場合、専用の記憶領域を作成し、複合ページ・フォーマット、SPAM 用のしきい値と間隔、ページ・サイズなどを設定して、ここにセグメント化文字列を格納することにより、パフォーマンスを向上できます。

さらに、統計から行が断片化していることがわかった場合は、SQL の ALTER DATABASE 文、EXPORT 文、IMPORT 文を使用してデータベース・パラメータを変更すれば、断片化を軽減できます。レコードの断片化を確認する方法や、断片化が大量に発生した場合の解決方法は、4.1.6 項（4-104）を参照してください。

表 4-8（4-117）は、Oracle Rdb で使用する記憶領域パラメータのデフォルト値です。

表 4-8 Oracle Rdb のマルチファイル記憶領域パラメータの値：デフォルト、最小、最大

記憶領域パラメータ	デフォルト値 （最小 / 最大）
記憶領域名	RDB$SYSTEM
記憶領域ファイル名	指定が必要（.rda）
割当て	400 ページ（1/ ディスク・デバイス・サイズ）
ページ・サイズ	2 ブロック（1 ブロック /32 ブロック）
ページ・フォーマット	Uniform
SPAM しきい値（複合フォーマット）	70%, 85%, 95%
SPAM しきい値（複合フォーマット）	0%, 0%, 0%
SPAM 間隔	216 ページ *1
記憶領域のエクステント・ページ	100 ページ（0/ ディスク・デバイス・サイズ）
記憶領域のエクステント・オプション	最小 99 ページ、最大 9,999 ページ、伸長 20%
スナップショット・ファイル名	記憶領域ファイルと同じ（.snp）
スナップショット・ファイルの割当て	100 ページ（0/ ディスク・デバイス・サイズ）
スナップショット・ファイルのエクステント・ページ	100 ページ（0/ ディスク・デバイス・サイズ）

表 4-8 Oracle Rdb のマルチファイル記憶領域パラメータの値：デフォルト、最小、最大（続き）

記憶領域パラメータ	デフォルト値 （最小 / 最大）
スナップショット・ファイルのエクステント・オプション	最小 99 ページ、最大 9,999 ページ、伸長 20%

*1(最小：216、最大：((1 ページあたりのブロック数 × 512) - 22) × 4)

データベースの変更が必要な場合は、SQL の ALTER DATABASE 文と CREATE STORAGE AREA 句を使用して、新しい記憶領域とファイル名で記憶領域を新しく定義してください。この方法で変更した方がよいのは、ページ・サイズ、SPAM の間隔としきい値、スナップショット・ファイル名、ページ・フォーマットです。新しい記憶領域を定義したら、SQL の ALTER STORAGE MAP 文の STORE 句を変更し、新しい記憶領域をポイントします。データは、古い記憶領域からアンロードされ、新しい記憶領域にロードされます。最後に、SQL の ALTER DATABASE 文で DROP STORAGE AREA 句を使用して、古い記憶領域を削除してください。

表 4-9（4-118）は、各 SQL 文について、SQL の CREATE DATABASE、ALTER DATABASE または IMPORT 文を使用して指定できる記憶領域パラメータを示しています。

表 4-9 記憶領域パラメータを変更する SQL 文

記憶領域パラメータ	SQL CREATE STORAGE AREA	SQL ALTER STORAGE AREA	SQL IMPORT または CREATE STORAGE AREA
記憶領域名の指定	可	可	可
記憶領域 RDB$SYSTEM の指定	可	不可	可
記憶領域ファイル名の指定	可	不可	可
割当て	可	不可	可
ページ・サイズ	可	不可	可
ページ・フォーマット	可	不可	可
SPAM しきい値の指定	可	不可	可
SPAM 間隔の指定	可	不可	可
記憶領域のエクステント・ページ	可	可	可
記憶領域のエクステント・オプション	可	可	可
スナップショット・ファイル名	可	不可	可
スナップショット・ファイルの割当て	可	可	可

表 4-9 記憶領域パラメータを変更する SQL 文（続き）

記憶領域パラメータ	SQL CREATE STORAGE AREA	SQL ALTER STORAGE AREA	SQL IMPORT または CREATE STORAGE AREA
スナップショット・ファイルのエクステント・ページ	可	可	可
スナップショット・ファイルのエクステント・オプション	可	可	可
読取り属性の指定	不可	可	不可

表 4-10（4-119）に、記憶領域パラメータを指定する Oracle の RMU コマンドを示します。

表 4-10 記憶領域パラメータを変更する Oracle の RMU コマンド

記憶領域パラメータ	RMU Restore	RMU Copy_Database	RMU Move_Area
記憶領域名の指定	不可	不可	不可
記憶領域 RDB$SYSTEM の指定	不可	不可	不可
記憶領域ファイル名の指定	可	可	可
割当て	不可	不可	不可
ページ・サイズ	可	可	可
ページ・フォーマット	不可	不可	不可
SPAM しきい値の指定	可	可	可
SPAM 間隔の指定	不可	不可	不可
記憶領域のエクステント・ページ	可	可	可
記憶領域のエクステント・オプション	不可	不可	不可
スナップショット・ファイル名の指定	可	可	可
スナップショット・ファイルの割当て	可	可	可
スナップショット・ファイルのエクステント・ページ	不可	不可	不可

表 4-10 記憶領域パラメータを変更する Oracle の RMU コマンド（続き）

記憶領域パラメータ	RMU Restore	RMU Copy_Database	RMU Move_Area
スナップショット・ファイルのエクステント・オプション	不可	不可	不可
読取り属性の指定	不可	不可	不可

RMU Restore、RMU Copy_Database および RMU Move_Area コマンドを使用した記憶領域の変更については、『Oracle Rdb7 Guide to Database Maintenance』と『Oracle RMU Reference Manual』を参照してください。

4.2.1 記憶領域パラメータ情報の収集

4.2.1.1 項（4-120）から 4.2.1.4 項（4-137）では、RMU Analyze コマンドと Performance Monitor の画面を使用した記憶領域情報の収集について説明します。RMU Analyz コマンドの使用方法に関する一般的な内容は、2.1 項（2-2）を参照してください。Performance Monitor の使用方法に関する一般的な内容は、2.2 項（2-15）を参照してください。

4.2.1.1 RMU Analyze Areas コマンド

この項では、RMU Analyze コマンドを使用し、Areas [= storage-area-list] および Option [= Normal, Full, Debug のいずれか] 修飾子を指定した場合の出力フォーマットと内容について説明します。

RMU Analyze Areas Option=Normal コマンドの使用方法

RMU Analyze Areas コマンドを使用し、Option=Normal 修飾子を指定して mf_personnel データベースに関する情報を収集すると、Oracle Rdb は、記憶領域と、記憶領域で定義されている各論理領域に関する重要な情報を概要で表示します。例 4-20（4-120）を参照してください。

例 4-20 RMU Analyze Areas コマンドで Option=Normal Qualifier を指定

```
$ RMU/ANALYZE/AREAS=EMPIDS_LOW/OPTION=NORMAL mf_personnel
------------------------------------------------------------------------------
 Storage analysis for storage area: EMPIDS_LOW - file: $DUA0:[ORION]EMPIDS_LOW.RDA;1
 Area_id: 2,  Page length: 1024,  Last page: 52
 Bytes free: 35992 (68%), bytes overhead: 6111 (11%)
 Spam count: 1, AIP count: 0, ABM count: 0
 Data records: 222, bytes used: 11145 (21%)
   average length: 50, compression ratio:  .87
   index records: 83, bytes used: 4768 (9%)
     B-Tree: 0, Hash: 1978, Duplicate: 2790, Overflow: 0
```

```
-----------------------------------
 Logical area: RDB$SYSTEM_RECORD for storage area : EMPIDS_LOW
 Larea id: 57,  Record type: 0, Record length: 215, Not Compressed
 Data records: 0, bytes used: 593 (1%)
-----------------------------------
 Logical area: EMPLOYEES_HASH for storage area : EMPIDS_LOW
 Larea id: 58,  Record type: 0, Record length: 215, Not Compressed
 Data records: 26, bytes used: 989 (2%)
   average length: 38
-----------------------------------
 Logical area: EMPLOYEES for storage area : EMPIDS_LOW
 Larea id: 63,  Record type: 26, Record length: 121, Compressed
 Data records: 37, bytes used: 2723 (5%)
   average length: 74, compression ratio:  .63
-----------------------------------
 Logical area: JOB_HISTORY_HASH for storage area : EMPIDS_LOW
 Larea id: 66,  Record type: 0, Record length: 215, Not Compressed
 Data records: 26, bytes used: 989 (2%)
   average length: 38
-----------------------------------
 Logical area: JOB_HISTORY for storage area : EMPIDS_LOW
 Larea id: 69,  Record type: 29, Record length: 42, Compressed
 Data records: 102, bytes used: 3654 (7%)
   average length: 36, compression ratio:  .97
--------------------------------------------------------------------------
```

例 4-20（4-120）で示すように、EMPIDS_LOW 記憶領域には、SQL の CREATE INDEX および CREATE STORAGE MAP 文の STORE 句を使用して、次の 4 つの論理領域が定義されています。

- EMPLOYEES_HASH ハッシュ・インデックス
- EMPLOYEES テーブル
- JOB_HISTORY_HASH ハッシュ・インデックス
- JOB_HISTORY テーブル

次に、EMPIDS_LOW 記憶領域に関する概要を一覧します。説明の後に続くカッコで囲んだ数字は、例 4-20（4-120）を参照してください。

- Area_id
  記憶領域を作成したときに割り当てた記憶領域 ID（2）
- Page length
  ページの長さをバイト単位で表示（1024）
- Last page
  記憶領域の最後のページ番号（52）

- Bytes free
  データの格納が可能な記憶領域内の総バイト数と空き領域のパーセンテージ（35,992 バイト、（68%）空き）
- Bytes overhead
  記憶領域で使用しているオーバーヘッドの総バイト数と、この値のパーセンテージ表示（6111 バイト（11%）使用済）
- Spam count
  記憶領域にある SPAM ページの合計（1）
- AIP count
  記憶領域にある AIP の合計（1）
- ABM count
  記憶領域にある ABM ページの合計（0）
- Data records
  すべての論理領域にあるデータ行の合計（222）。これは、次の数値の合計です。
  - 記憶領域内にあるすべての論理領域の "Data records" カウントの合計
  - 記憶領域内にあるすべての論理領域について、すべてのハッシュ・インデックスの重複ノード・レコードの数
  - 記憶領域内のすべての論理領域について、すべてのソート・インデックスのインデックス・ルートの下にあるインデックス・ノード（ランク付きインデックスの重複ノードと、ランクなしインデックスのオーバーフロー・ノード）の数
- Bytes used
  記憶領域で使用している総バイト数と、この値のパーセンテージ表示（11,145 バイト、（21%）使用済）
- Average length
  記憶領域にあるすべてのレコードの行の長さの平均（50）
- Compression ratio
  記憶領域にあるすべての行の平均圧縮率（圧縮 / 非圧縮）（0.87）
- Index records
  記憶領域にあるすべてのインデックスのインデックス・レコードの合計（83）。ハッシュ・インデックスについては、この値はハッシュ・バケットと重複ノード・レコードの数になります。ソート・インデックスについては、ルート・ノードの下にあるインデックス・ノード（ランクなしインデックスの重複ノードと、ランク付きインデックスのオーバーフロー・ノードを含む）の数になります。
- Bytes used
  すべてのインデックス・レコードで使用している総バイト数と、記憶領域に対するパーセンテージ表示（4768 バイト（9%）使用済）

- B-Tree, Hash, Duplicate, Overflow
  各レコード・タイプで使用している総バイト数（B-tree：0、Hash：1978、Duplicates：2790、Overflow：0）

次に、EMPLOYEES_HASH 論理領域に関する情報を一覧します。説明の後に続くカッコで囲んだ数字については、例 4-20（4-120）を参照してください。

- Logical area name
  論理領域の名前（EMPLOYEES_HASH）
- 記憶領域のストレージ分析
  記憶領域名（EMPIDS_LOW）
- Larea id
  論理領域を作成したときに割り当てた ID 番号（58）
- Record type
  テーブル ID、インデックスは常に 0（0）
- Record length
  最大レコード長（非圧縮）に、このレコード・タイプのオーバーヘッドを加算（215）。この値は、複合記憶領域のインデックスを記述する論理領域では無効です。
- Compression setting
  レコードの圧縮。レコードを圧縮しているかしていないかを示します（Not compressed）。
- Data records
  論理領域にあるデータ・レコードの総数（26）。この合計には、ハッシュ・インデックスのハッシュ・バケット、ソート・インデックスのルート・ノード、テーブルのデータ行が含まれます。ハッシュ・インデックスについては、この値には重複ノード・レコードは含まれません。ソート・インデックスについては、インデックス・ルートの下にあるインデックス・ノード（重複ノードとオーバーフロー・ノードを含む）は含まれません。
- Bytes used
  ストレージの総バイト数と、論理領域にあるすべてのデータ行が使用している領域（ハッシュ・インデックスのハッシュ・バケット、ソート・インデックスのルート・ノード、テーブルのデータ行）のパーセンテージ表示。この値には、オーバーヘッドは含まれません。Option=Debug 修飾子を指定した場合に出力表示されるレコード別の「SIZE」と同じ値です（989 バイト、（2%）使用済）。
- Average length
  圧縮または非圧縮レコードの平均の長さ（ハッシュ・バケット、ルート・ノード、テーブル・データ行）（38）
- Compression ratio
  すべてのレコードでの平均圧縮率（圧縮 / 非圧縮）。この統計は、テーブルのみで表示されます。ハッシュ・インデックスのレコードは圧縮されないため、表示されません。

例 4-20（4-120）では、EMPIDS_LOW 記憶領域に関して重要な統計がいくつか表示されているので、注意してください。

- 「Bytes free」と記憶領域の空きパーセンテージ
  記憶領域の空き領域は、35,992 バイト、68% であり、ここにレコードを追加で格納できます。平均レコード長（50 バイト）をもとに、記憶領域に追加可能なレコード数の推定値を計算できます（719 レコードが追加可能）。これは、簡単な式で計算した相対的な推測値です。さらに正確な値を計算するには、論理領域にある各レコードの平均レコード長を元に、レコード数を推測します。
  行をクラスタ化している場合は、親と子両方の行を同じページに格納しているため、ハッシュ・バケットのエントリが発生し、ハッシュ・バケットのエントリの数が多くなっていると考えられるため、使用可能なストレージの計算は複雑になります。各親行の重複数と、追加する親行の数がわかれば、テーブルに追加で格納できる親および子の行を計算できます。ハッシュ構造と、ハッシュ構造が使用する領域も考慮する必要があります。このような情報は、後で説明する Option=Debug 修飾子でわかります。
  Option=Full 修飾子（この項の後で説明）で表示されるヒストグラムから、データベースが増大していく様子がわかります。
  データベースの運用中に何回か分析を実行し、空きバイト数のパーセンテージと、使用済バイト数にオーバーヘッド・バイト数を加算したパーセンテージを比較することにより、データベースのサイズの増加や変化の様子がわかります。

- Data records、bytes used、byes overhead、パーセンテージ表示は、次のような内容を示します。
  記憶領域には合計で 222 のデータ行があり、記憶領域の 21% の領域を使用しています。各論理領域について、レコード数をカウントすると、次のような情報がわかります。

  - EMPLOYEES では、37 行が 2723 バイト（5%）を使用し、JOB_HISTORY では、102 行が 3654 バイト（7%）を使用しているので、合計で記憶領域の 12% が使用済です。

  - 各ハッシュ・インデックス論理領域（EMPLOYEES_HASH と JOB_HISTORY_HASH）にはそれぞれ 26 のハッシュ・バケットがあり、合計で 52 のハッシュ・バケットが 1978 バイトを使用しています。インデックス・レコードは 83 あります。この差（83-52=31）は、重複ノード・レコードの数であり、2790 バイトを使用しています。両方のハッシュ構造は、4768 バイト、つまり記憶領域の 9% を使用しています。

  - オーバーヘッドは、6111 バイト、つまり記憶領域の 11% を使用しています。データ行（12%）、インデックス・レコード（9%）、オーバーヘッド（11%）を合計すると、記憶領域の 32% を使用していることがわかります。

上記の数値から、論理領域とオーバーヘッドについて、使用領域比率をさらに計算できます。この値は、記憶領域内で、各論理領域やオーバーヘッドが使用している領域を、相対的な比率で表したものです。レコードの特性や配置がほとんど同じであるという前提に基づいて、記憶領域内で論理領域が使用している領域を、この統計情報から推測できます。定期的に分析を実行し、結果に基づいてこの統計を計算しておけば、各論理領域について、レコードの特性や配置の変化を把握できます。

使用領域比率は、次のように、各論理領域とオーバーヘッドについて計算します。各論理領域の使用済およびオーバーヘッドのパーセンテージを比較してください。この例では、5:2:7:2:11 です。これは、EMPLOYEES データ行の 37（5%）、EMPLOYEES_HASH ハッシュ・インデックス・レコードの 26（2%）、JOB_HISTORY データ行の 102（7%）、JOB_HISTORY_HASH のハッシュ・インデックス・レコードの 28（2%）、オーバーヘッド（11%）の比率を順番に表しています。このような比較から、使用領域比率は次のようになります。

- EMPLOYEES データ行と EMPLOYEES_HASH ハッシュ・バケットは、2.5:1
- JOB_HISTORY データ行と JOB_HISTORY_HASH ハッシュ・バケットおよび重複ノード・レコードは、3.5:1
- EMPLOYEES および EMPLOYEES_HASH データ行と JOB_HISTORY および JOB_HISTORY_HASH データ行は、0.78:1
- すべての記憶領域レコードとオーバーヘッドは、1.45:1

この比率は、レコードの特性や配置に大きな変化がなければ、将来的な領域の使用状況を予測するのに便利です。これは、相対的な比率なので、mf_personnel データベースのようにデータベースのサイズが小さな場合、オーバーヘッドのサイズは大きく見えます。データベースのサイズが大きくなっても、オーバーヘッドの使用領域は、データベースの増加率ほど大きくなりません。RMU Analyze コマンドを Areas 修飾子を指定して定期的に実行し、データベースの記憶領域に含まれるすべての論理領域について、オーバーヘッド領域の増加と使用済領域の増加の比率を計算してください。

RMU Analyze Areas Option=Full コマンドの使用方法

RMU Analyze Areas コマンドを使用し、Option=Full 修飾子を指定して mf_personnel データベースに関する情報を収集すると、Oracle Rdb では、Option=Normal 修飾子で表示されるフォーマットと内容の情報に加えて、次の 3 種類のヒストグラムが表示されます。

- ページごとの記憶領域のページ領域使用率のヒストグラム
  このヒストグラムは、記憶領域にあるページ数の合計に対して、すべての論理領域のデータに使用するページ領域のパーセンテージを示します。ページ使用率は、十分位数に分類されます。カッコ内の値は、十分位数の分類に該当するページ数合計です。
- 論理領域のページ領域使用率とページ数のヒストグラム
  このヒストグラムは、データを格納するページの数に対して、特定の論理領域のデータに使用するページ領域のパーセンテージ（(used/used+free) × 100）を示します。ページ使用率は、十分位数に分類します。カッコ内の値は、十分位数の分類に該当するページ数合計です。
- 最大レコード長に対する論理領域のパーセンテージとレコード数のヒストグラム
  このヒストグラムは、各レコードのレコード型の最大サイズ（非圧縮）に対するパーセンテージを示します。最大レコード長のパーセンテージは、十分位数に分類されます。カッコ内の値は、十分位数の分類に該当するレコード総数です。

RMU Analyze Areas コマンドで Option=Full 修飾子を指定して実行し、EMPIDS_LOW 記憶領域を分析すると、Oracle RMU は次のような情報を表示します。

- 概要情報。
- EMPIDS_LOW 記憶領域について、記憶領域のページ使用率とページ数のヒストグラム（% used vs # pages）を表示します。
- 論理領域の情報。
- EMPLOYEES_HASH ハッシュ・インデックス、EMPLOYEES テーブル、JOB_HISTORY_HASH ハッシュ・インデックス、JOB_HISTORY テーブルの 4 つの論理領域について、2 つの論理領域ヒストグラム（used/used+free vs # pages と % of max length vs # records）が表示されます。

表示内容は、例 4-21（4-126）を参照してください。

例 4-21 Option=Full 修飾子を使用した RMU Analyze Areas コマンド

```
$ RMU/ANALYZE/AREAS=EMPIDS_LOW/OPTION=FULL mf_personnel
-------------------------------------------------------------------------------
 Storage analysis for storage area: EMPIDS_LOW - file: $DUA0:[ORION]EMPIDS_LOW.RDA;1
 Area_id: 2,  Page length: 1024,  Last page: 52
 Bytes free: 35992 (68%), bytes overhead: 6111 (11%)
 Spam count: 1, AIP count: 0, ABM count: 0
 Data records: 222, bytes used: 11145 (21%)
   average length: 50, compression ratio:  .87
   index records: 83, bytes used: 4768 (9%)
     B-Tree: 0, Hash: 1978, Duplicate: 2790, Overflow: 0
                               % used vs # pages
  >90% | (0)
80-90% | (0)
70-80% |======== (4)
60-70% |== (1)
50-60% |====== (3)
40-50% |== (1)
30-40% |=================== (10)
20-30% |=========== (6)
10-20% |== (1)
 0-10% |====================================================== (26)
---------------------------------
 Logical area: RDB$SYSTEM_RECORD for storage area : EMPIDS_LOW
 Larea id: 57,  Record type: 0, Record length: 215, Not Compressed
 Data records: 0, bytes used: 593 (1%)
                           used/used+free vs # pages
  >90% | (0)
80-90% | (0)
70-80% | (0)
```

```
60-70% | (0)
50-60% |= (1)
40-50% | (0)
30-40% | (0)
20-30% |== (2)
10-20% |= (1)
 0-10% |================================================== (47)
                        % of max length vs # records
  >90% | (0)
80-90% | (0)
70-80% | (0)
60-70% | (0)
50-60% | (0)
40-50% | (0)
30-40% | (0)
20-30% | (0)
10-20% | (0)
 0-10% |================================================== (51)
-----------------------------------
 Logical area: EMPLOYEES_HASH for storage area : EMPIDS_LOW
 Larea id: 58,  Record type: 0, Record length: 215, Not Compressed
 Data records: 26, bytes used: 989 (2%)
   average length: 38
                          used/used+free vs # pages
  >90% | (0)
80-90% |== (1)
70-80% | (0)
60-70% |== (1)
50-60% |== (1)
40-50% |== (1)
30-40% | (0)
20-30% | (0)
10-20% |===== (2)
 0-10% |================================================== (20)
                        % of max length vs # records
  >90% | (0)
80-90% | (0)
70-80% | (0)
60-70% | (0)
50-60% | (0)
40-50% | (0)
30-40% |== (1)
20-30% |===== (2)
10-20% |=========== (4)
 0-10% |================================================== (19)
-----------------------------------
 Logical area: EMPLOYEES for storage area : EMPIDS_LOW
```

```
 Larea id: 63,  Record type: 26, Record length: 121, Compressed
 Data records: 37, bytes used: 2723 (5%)
   average length: 74, compression ratio:  .63
                           used/used+free vs # pages
  >90% |=== (1)
80-90% |======= (2)
70-80% | (0)
60-70% |=== (1)
50-60% | (0)
40-50% | (0)
30-40% |=== (1)
20-30% |======= (2)
10-20% |============== (4)
 0-10% |================================================== (15)
                          % of max length vs # records
  >90% | (0)
80-90% | (0)
70-80% | (0)
60-70% |========= (5)
50-60% |================================================== (30)
40-50% |=== (2)
30-40% | (0)
20-30% | (0)
10-20% | (0)
 0-10% | (0)
-----------------------------------
 Logical area: JOB_HISTORY_HASH for storage area : EMPIDS_LOW
 Larea id: 66,  Record type: 0, Record length: 215, Not Compressed
 Data records: 26, bytes used: 989 (2%)
   average length: 38
                           used/used+free vs # pages
  >90% | (0)
80-90% |== (1)
70-80% | (0)
60-70% |== (1)
50-60% |== (1)
40-50% |== (1)
30-40% | (0)
20-30% | (0)
10-20% |===== (2)
 0-10% |================================================== (20)
                          % of max length vs # records
  >90% | (0)
80-90% | (0)
70-80% | (0)
60-70% | (0)
50-60% | (0)
```

```
40-50% | (0)
30-40% |== (1)
20-30% |===== (2)
10-20% |=========== (4)
 0-10% |================================================== (19)
-----------------------------------
 Logical area: JOB_HISTORY for storage area : EMPIDS_LOW
 Larea id: 69,  Record type: 29, Record length: 42, Compressed
 Data records: 102, bytes used: 3654 (7%)
   average length: 36, compression ratio:  .97
                          used/used+free vs # pages
  >90% |======= (1)
80-90% |=============== (2)
70-80% |======= (1)
60-70% |======= (1)
50-60% | (0)
40-50% |=============== (2)
30-40% |=============== (2)
20-30% |====================== (3)
10-20% |================================================== (7)
 0-10% |================================================== (7)
                         % of max length vs # records
  >90% | (0)
80-90% |================================================== (65)
70-80% |============================== (37)
60-70% | (0)
50-60% | (0)
40-50% | (0)
30-40% | (0)
20-30% | (0)
10-20% | (0)
 0-10% | (0)
--------------------------------------------------------------------------------
```

記憶領域のページ領域使用率とページ数のヒストグラム（% used vs # pages）は、すべての論理領域のデータに使用するページ領域を、記憶領域にあるページ数の合計に対するパーセンテージで示したものです。ヒストグラムの形状は、アプリケーションの種類やデータベースの履歴によって異なります。CREATE STORAGE MAP 文で PLACEMENT VIA 句を指定しないでデータベースをロードした場合、ほとんどのページは、ヒストグラムの一番上または一番下に分類されます。Oracle Rdb は、ページがいっぱいになるまでデータベース・ページにデータを格納してから、次のページに進みます。すべてのデータを格納しても、一部は空きページとして残ります。

EMPIDS_LOW 記憶領域では、ほとんどのページが、ヒストグラムの一番下に分類されています。この記憶領域に含まれる EMPLOYEES 行と JOB_HISTORY 行は、PLACEMENT VIA

INDEX 句で配置され、EMPLOYEES 行は EMPLOYEES_HASH ハッシュ・インデックス、JOB_HISTORY 行は JOB_HISTORY_HASH を使用します。

さらに、used/used+free vs # pages と % of max length vs # records の 2 つの論理領域ヒストグラムを使えば、次のような分析ができます。

- EMPLOYEES_HASH と JOB_HISTORY_HASH 論理領域のレコードは、圧縮されていません。各インデックスのハッシュ・バケットのサイズは、平均 38 バイトです。ほとんどのレコード・サイズ（EMPLOYEES_HASH と JOB_HISTORY_HASH の両方で、26 のうちの 19）は、最大長の 10% 程度です。ハッシュ・バケットは、52 ページのうち 26 ページに分散しており、ハッシュ・バケットが格納されているページのほとんどには十分な空き領域があるので、ハッシュ・バケットの増大や新規レコードの追加にも対応できます。
- EMPLOYEES テーブルでは、行を圧縮しているので、平均レコード長は 74 バイト、平均圧縮率は 0.63 です。ほとんどの行（37 のうち 32）のサイズは、最大長の 60% 未満です。37 の EMPLOYEES 行が、26 ページに分散しています。EMPLOYEES 行が格納されているページのほとんど（26 ページのうち 23）に十分な空き領域があり、EMPLOYEES 行の追加ができます。
- JOB_HISTORY テーブルでは、行を圧縮し、平均レコード長は 36 バイト、平均圧縮率は 0.97 です。行の半分以上（102 のうち 65）が、最大レコード長の 80 ～ 90% です。合計で 102 行が 26 ページに分散しています。JOB_HISTORY 行が格納されているページのほとんどには（26 ページのうち 22）十分な空き領域があり、JOB_HISTORY 行の追加ができます。

データベースを頻繁に更新する環境では、使用率が 100% に近いページが多くなります。頻繁に更新すると、行が断片化します。これにより、パフォーマンスは低下します。

RMU Analyze Areas コマンドを実行し、使用済領域が 60% を超えるページが多数存在する場合は、データベース・パフォーマンスを調べてください。一般的に、記憶領域では、30 ～ 50% を空き領域として定義してください。

データベース・パフォーマンスが低く、RMU Analyze Areas を実行した結果、未使用のページや断片化した行が多数存在することがわかった場合は、SQL の ALTER DATABASE 文を実行して新しい記憶領域を定義して、この記憶領域に最適なパラメータを設定し、ストレージ・マップの変更によって古い記憶領域から新しい記憶領域にデータをロードします。詳細な手順は、『Oracle Rdb7 Guide to Database Maintenance』を参照してください。

RMU Analyze Areas Option=Debug コマンドの使用方法

RMU Analyze Areas コマンドで Option=Debug 修飾子を指定して実行し、EMPIDS_LOW 記憶領域の情報を収集すると、Oracle Rdb は、次のように 2 つの部分で構成される内容を表示します。

- 最初の部分では、記憶領域に含まれる各ページの詳細なアカウンティング情報を示します。この部分は、次のような順序で表示されます。

- 各論理領域の概要情報
- ページ上にある 11 種類のレコード型を定義する凡例
- 各記憶領域のページに関する詳細なアカウンティング情報

- 2 番目の部分には、Option=Full 修飾子を指定した場合と同じ内容が表示されます。出力フォーマットと内容については、この項の RMU Analyze Areas コマンドの Option=Full 修飾子の説明を参照してください。

例 4-22（4-131）に、RMU Analyze Areas コマンドで Option=Debug 修飾子を指定した場合の出力内容を示します。

例 4-22 Option=Debug 修飾子を使用した RMU Analyze Areas コマンド

```
$ RMU/ANALYZE/AREAS=EMPIDS_LOW/OPTION=DEBUG mf_personnel
0------------------------------------------------------------------------------
0
0 Storage-area-name              Area-ID Page-len Last-page File-name
0
0   Logical-area-name              Larea-ID Rec-type    Rec-len Compressed
0------------------------------------------------------------------------------
0
1 EMPIDS_LOW                             2    1024         52
$DUA0:[ORION]EMPIDS_LOW.RDA;1
0
2   RDB$SYSTEM_RECORD                   57       0        215   F
2   EMPLOYEES_HASH                      58       0        215   F
2   EMPLOYEES                           63      26        121   T
2   JOB_HISTORY_HASH                    66       0        215   F
2   JOB_HISTORY                         69      29         42   T
0
0
0------------------------------------------------------------------------------
0
0 TYPES:  0 SPAM              4 B-tree index      8 First data fragment
0         1 AIP               5 Hash index        9 First fragment compressed
0         2 ABM               6 data record      10 Next data fragment
0         3 System record     7 Compressed data  11 Next fragment compressed
0
0 AREA    LAREA   PNO        FREE    SIZE    EXPANDED TYPE TOTAL-SIZE
0------------------------------------------------------------------------------
0
3    2       0          1     948      54          54   0        54
3    2      57          2      62      18          18   3        18
3    2      63          2      62      71         116   7        71
3    2      58          2      62      49          49   5        49
3    2      63          2      62      73         116   7        73
```

```
3    2      69          2      62      38          37   7        38
3    2      66          2      62      49          49   5        49
3    2      69          2      62      38          37   7        38
3    2       0          2      62      90          90   5        90
3    2      69          2      62      38          37   7        38
3    2      69          2      62      32          37   7        32
3    2      69          2      62      38          37   7        38
3    2      69          2      62      38          37   7        38
3    2       0          2      62      90          90   5        90
3    2      69          2      62      32          37   7        32
3    2      69          2      62      38          37   7        38
3    2      57          3     964       5           5   3         5
3    2      57          4     964       5           5   3         5
3    2      57          5     344      18          18   3        18
3    2      63          5     344      73         116   7        73
3    2      58          5     344      30          30   5        30
3    2      69          5     344      38          37   7        38
3    2      69          5     344      38          37   7        38
3    2      69          5     344      32          37   7        32
3    2      69          5     344      32          37   7        32
.
.
.
```

Option=Debug 修飾子を指定して RMU Analyze Areas=EMPIDS_LOW コマンドを実行した場合の出力フォーマットと内容は、次のように、ヘッダーで分類されています（詳細な出力セクションでは、EMPLOYEES_HASH 論理領域と最初の行（PNO 1）について表示）。

- メイン・ヘッダー
  - Storage-area-name
    記憶領域名（EMPIDS_LOW）
  - Area-ID
    記憶領域 ID（2）
  - Page-len
    記憶領域のページの長さ（1024）
  - Last-page
    記憶領域の最後のページ番号（52）
  - File-name
    記憶領域のファイル名（$DUA0:[ORION]empids_low.rda;1）
- 2 つ目のヘッダー（EMPLOYEES_HASH 論理領域についてのみ表示）
  - Logical-area-name
    論理領域名（EMPLOYEES_HASH）

- Larea-id
  記憶領域 ID（58）
- Rec-type
  記憶領域のレコード・タイプ（テーブル ID）。インデックスは常に 0（0）
- Rec-len
  最大レコード長（非圧縮）に、このレコード型プのオーバーヘッドを加算（215）
  この値は、複合記憶領域の論理領域では無効です。
- Compressed
  レコード圧縮。コード化された値で、「T」は圧縮、「F」は非圧縮（F）。

- レコード・タイプの凡例
  ページ上にある 11 種類のレコード型を定義する凡例。この値は、詳細な出力セクションの「TYPE」の欄に表示されます。
- 詳細な記憶領域ページの内容（最初の行は、PNO=1）
  - AREA
    記憶領域 ID 番号（2）
  - LAREA
    論理領域 ID 番号（0）
  - PNO
    記憶領域ページ番号。ページ上にあるすべての論理領域 ID は、ページ番号でグループ化します（1）。
  - FREE
    そのページ上の論理領域にある空きバイト数（948）
  - SIZE
    論理領域にある実際のレコード・サイズ（バイト単位）で、圧縮または非圧縮（54）
  - EXPANDED
    レコードを圧縮している場合は、復元したレコード・サイズ（バイト単位）（54）。
  - TYPE
    コード化したレコード型。出力テーブルの「TYPES」で表示されている凡例を参照してください（0 は、SPAM ページ・レコードを示す）。
  - TOTAL-SIZE
    論理領域内のレコードのすべての断片を含む、実際のレコード長（54）。
- Option=Full の情報
  出力内容については、この項で説明した RMU Analyze Areas Option=Full コマンドを参照してください。

例 4-22（4-131）の「TYPE」に表示されている内容をさらに見ると、どのテーブルのどのデータ・ページにも、断片化された行（TYPE = 8、9、10 または 11）はないことがわかります。この情報は、表示内容の中で最も重要です。ページ上で使用可能な空き領域よりも行が大きくなった場合や、最初に格納したデータベース・ページよりも大きくなった場合に、行は断片化します。断片化した行が多数存在する場合、次のようにいくつかの解決方法があります。

- SQL の ALTER DATABASE 文を使用して新しい記憶領域を定義し、最適なパラメータを設定（ページ・サイズの拡張など）または SPAM しきい値に小さな値を設定して、ストレージ・マップの変更により古い記憶領域から新しい記憶領域にデータをロードします。詳細な手順は、『Oracle Rdb7 Guide to Database Maintenance』を参照してください。
- SQL の EXPORT 文と IMPORT 文を使用して、ページ・サイズの大きなデータベースを新しく作成するか、ページ・サイズが十分大きい場合には、SPAM しきい値に小さな値を設定します。

いくつかの記憶領域で断片化が発生している場合は、SQL の ALTER DATABASE 文を使うことをお薦めします。

ページ上の各論理領域を調べると、ページ上にある論理領域、論理領域のレコード・サイズ、空き領域の容量などがわかります。たとえば、例 4-22（4-131）のページ 2（PNO 2）からは、4 つの異なる論理領域とシステム・レコードが 824 バイト使用しているので、ページ上の空き領域は残り 62 バイトだということがわかります。別の EMPLOYEES の親行を、JOB_HISTORY の子行と一緒に追加することはできないので、このページはいっぱいだとみなされます。ページ 2 には、次のようなレコードが格納されています。

- 2 つの EMPLOYEES 行（各 116 バイト、合計 232 バイト）
- 8 つの JOB_HISTORY 行（各 37 バイト、合計 296 バイト）
- 1 つの SYSTEM レコード（18 バイト）
- 2 つのハッシュ・バケット（各 49 バイト、合計 98 バイト）
- 2 つの重複ノード・レコード（各 90 バイト、合計 180 バイト）

このような方法でページの内容を分析すると、記憶領域のページがどのように使用されているかわかります。この情報とヒストグラムを組み合せることによって、問題の早期発見が可能になります。データベースを頻繁に更新し、サイズが増大するアプリケーション環境では、定期的にデータベースを分析してください。目的のページ範囲がわかっている場合は、Start および End 修飾子を使用して、必要なページ範囲のみを分析できます。

また、Option=Debug 修飾子を指定した RMU Analyze Area コマンドからの出力を、ユーザー設計のプログラムの入力として使用すれば、カスタマイズした概要レポートを作成できます。たとえば、論理領域やページごとに情報をまとめることもできます。

4.2.1.2 「RMU Analyze Lareas」表示

この項では、RMU Analyze コマンドを、Lareas および Option 修飾子を指定して実行した場合の出力フォーマットと内容について説明します。

RMU Analyze Lareas Option=Normal コマンドの使用方法

RMU Analyze コマンドを使用して、Lareas および Option=Normal 修飾子を指定すると、Oracle Rdb は、各記憶領域で定義されているすべての論理領域について、重要な情報を概要で表示します。表示されるフォーマットや内容は、Option=Normal 修飾子を指定した RMU Analyze コマンドとまったく同じです。詳細は、4.2.1.1 項（4-120）で説明している RMU Analyze Areas Option=Normal コマンドの使用方法を参照してください。

このコマンドは、Lareas 修飾子と Areas 修飾子を指定することによって、表示する記憶領域と論理領域を指定できます。RDB$SYSTEM 記憶領域内に、同じ名前と論理 ID 番号を持つ論理領域が存在しても、指定しなければ表示されないので注意してください。修飾子 Areas=DEPARTMENTS、Lareas=(DEPARTMENTS、DEPARTMENTS_INDEX) および Option=Normal を指定して RMU Analyze コマンドを実行すると、例 4-23（4-135）に示すように、指定した領域のみの情報が表示されます。

例 4-23 修飾子 Lareas および Option=Normal を指定した RMU Analyze Areas コマンド

```
$ RMU/ANALYZE/AREAS=DEPARTMENTS /LAREAS=(DEPARTMENTS,DEPARTMENTS_INDEX) -
_$ /OPTION=NORMAL mf_personnel
-------------------------------------------------------------------------------
 Storage analysis for storage area: DEPARTMENTS - file:
$DUA0:[ORION]DEPARTMENTS.RDA;1
 Area_id: 5,  Page length: 1024,  Last page: 28
 Bytes free: 25144 (88%), bytes overhead: 2062 (7%)
 Spam count: 1, AIP count: 0, ABM count: 0
 Data records: 27, bytes used: 1466 (5%)
   average length: 54, compression ratio:  .85
   index records: 1, bytes used: 428 (1%)
     B-Tree: 428, Hash: 0, Duplicate: 0, Overflow: 0
-----------------------------------
 Logical area: DEPARTMENTS_INDEX for storage area : DEPARTMENTS
 Larea id: 72,  Record type: 0, Record length: 215, Not Compressed
 Data records: 1, bytes used: 428 (1%)
   average length: 428
-----------------------------------
 Logical area: DEPARTMENTS for storage area : DEPARTMENTS
 Larea id: 73,  Record type: 28, Record length: 55, Compressed
 Data records: 26, bytes used: 1038 (4%)
   average length: 40, compression ratio:  .80
-------------------------------------------------------------------------------
```

DEPARTMENTS 論理領域は、RDB$SYSTEM 記憶領域内に存在していますが、指定されていないので表示されない点に注意してください。

RMU Analyze Lareas Option=Full コマンドの使用方法

Option=Full 修飾子を指定して RMU Analyze Lareas コマンドを実行すると、Option=Normal 修飾子を指定して RMU Analyze Lareas コマンドを実行した場合と同じ情報が表示されます。さらに、Areas と Option=Full 修飾子を指定して RMU Analyze コマンドを実行した場合と同じ 3 種類のヒストグラムも表示されます。詳細は、4.2.1.1 項（4-120）を参照してください。

Lareas および Option=Full 修飾子を指定すると、論理領域に関する情報が次の順序で表示されます。

- すべての論理領域に関する概要情報と概要のヒストグラムを、記憶領域ごとに表示
- 各論理領域と記憶領域のページ領域使用率に関する情報を、ページ数のヒストグラムで表示
- 各記憶領域の各論理領域について、ヒストグラムをペアで表示

RMU Analyze Lareas Option=Debug コマンドの使用方法

Option=Debug 修飾子を指定して RMU Analyze Lareas コマンドを実行すると、4.2.1.1 項（4-120）で説明した RMU Analyze Areas Option=Debug と同様に、情報は 2 つの部分に分かれて表示されます。ただし、選択した論理領域とシステム・レコード（論理領域 ID は 0）の情報のみが表示されます。

- 最初の部分では、記憶領域に含まれる各ページの詳細なアカウンティング情報を示します。この部分は、次のような順序で表示されます。
  - 各論理領域の概要情報
  - ページ上にある 11 種類のレコード型を定義する凡例
  - 各記憶領域のページに関する詳細なアカウンティング情報
- 2 番目の部分には、Option=Full 修飾子を指定した場合と同じ内容が表示されます。出力フォーマットと内容については、4.2.1.1 項（4-120）の RMU Analyze Areas コマンドの Option=Full 修飾子の説明を参照してください。

表示内容と例については、4.2.1.1 項（4-120）の Option=Debug を指定した RMU Analyze Areas コマンドの説明を参照してください。

4.2.1.3 Performance Monitor の「Storage Area Information」画面

Oracle Rdb では、データベースの各記憶領域に関する情報を表示できます。「Database Parameter Information」サブメニューから「Storage Area Information」オプションを選択してください。次に、「Storage Area Information」画面の例を示します。

```
Node: TRIXIE         Oracle Rdb V7.0-00 Performance Monitor 28-MAY-1996 13:57:59
Rate: 3.00 Seconds          Storage Area Information        Elapsed: 03:37:05.72
Page: 1 of 60      RDBVMS_USER1:[LOGAN.V70]MF_PERSONNEL.RDB;1       Mode: Online
--------------------------------------------------------------------------------
Storage area "RDB$SYSTEM"
Area ID number is 1
Filename is "RDBVMS_USER1:[LOGAN.V70]MF_PERS_DEFAULT.RDA;1"
Access mode is read/write
Page format is uniform
Page size is 2 blocks
- Current physical page count is 1036
Row level locking is enabled
Row caching is disabled
No row cache is defined for this area
Extends are enabled
- Extend area by 20%, minimum of 99 pages, maximum of 9999 pages
- Area has been extended 8 times
Volume set spreading is enabled
Snapshot area ID number is 31
SPAMs are enabled
- Interval is 1089 data pages
--------------------------------------------------------------------------------
Exit Help Menu >next_page <prev_page Options Refresh Set_rate Write !
```

画面の詳細は、Performance Monitor のヘルプを参照してください。

4.2.1.4 Performance Monitor の「I/O Statistics」画面

Oracle Rdb では、データベースの各ファイルに関する I/O 統計を表示できます。Performance Monitor の「IO Statistics (by file)」を選択すると、Oracle Rdb は、データベースを構成するファイルのメニューを表示するので、そのファイルの統計を表示できます。次に、「I/O Statistics (by file)」メニューの例を示します。

```
Node: ALPHA3                                                  14-MAR-1995 14:18:24
Rate: 3.00 Seconds                Select File                 Elapsed: 00:05:54.89
Page: 1 of 1    RDBVMS +-------------------------------+  .RDB;1    Mode: Online
---------------------- |  A. File IO Overview          | ----------------------
statistic.........     |  B. Device IO Overview        | ....... average......
name..............     |  C. Device Information        | ....... per.trans....
transactions           |  D. root file                 |       0           0.0
verb successes         |  E. AIJ file                  |       0           0.0
verb failures          |  F. RUJ file                  |       0           0.0
                       |  G. ACE file                  |
synch data reads       |  H. all data/snap files       |       0           0.0
synch data writes      |  I. data file MF_PERS_DEFAULT |       0           0.0
asynch data reads      |  J. data file EMPIDS_LOW      |       0           0.0
asynch data writes     |  K. data file EMPIDS_MID      |       0           0.0
RUJ file reads         |  L. data file EMPIDS_OVER     |       0           0.0
RUJ file writes        |  M. data file DEPARTMENTS     |       0           0.0
AIJ file reads         |  N. data file SALARY_HISTORY  |       0           0.0
AIJ file writes        |  O. data file JOBS            |       0           0.0
ACE file reads         |  P. data file EMP_INFO        |       0           0.0
ACE file writes        |  Q. <<more>>                  |       0           0.0
root file reads        |                               |      17           0.0
root file writes       +-------------------------------+       0           0.0
------------------------------------------------------------------------------
Type <return> or <letter> to select file, <control-Z> to cancel menu
```

すべての .rda および .snp ファイルが表示されます。この画面は、グラフ形式では表示できません。画面には、Performance Monitor セッションを開始した後、または「Reset」オプションでアキュムレータを最後にリセットした後の情報が適用されます。

画面の各フィールドについては、Performance Monitor のヘルプを参照してください。

次に、「File I/O Statistics」画面で、すべてのデータおよびスナップショット・ファイルを表示する例を示します。

```
Node: TRIXIE         Oracle Rdb V7.0-00 Performance Monitor 28-MAY-1996 12:52:53
Rate: 3.00 Seconds             File IO Statistics          Elapsed: 02:31:59.67
Page: 1 of 1       RDBVMS_USER1:[LOGAN.V70]MF_PERSONNEL.RDB;1       Mode: Online
-------------------------------------------------------------------------------
                          For File: All data/snap files
statistic........... rate.per.second.............. total....... average......
name................ max..... cur..... avg....... count....... per.trans....
total I/Os                          0                      36          12.0
    (Synch. reads)          0       0        0.0           36          12.0
    (Synch. writes)         0       0        0.0            0           0.0
    (Extends)               0       0        0.0            0           0.0
    (Asynch. reads)         0       0        0.0            0           0.0
    (Asynch. writes)        0       0        0.0            0           0.0
statistic........... blocks.transferred.........  stall.time.(x100)...........
name................ avg.per.I/O.. total........  avg.per.I/O... total........
total I/Os                   6.0          216             0.6            25
    (Synch. reads)           6.0          216             0.6            25
    (Synch. writes)          0.0            0             0.0             0
    (Extends)                0.0            0             0.0             0
    (Asynch. reads)          0.0            0             0.0             0
    (Asynch. writes)         0.0            0             0.0             0
-------------------------------------------------------------------------------
Exit Help Menu Options Reset Set_rate Unreset Write !
```

OpenVMS VAX OpenVMS Alpha

「Device IO Overview」画面では、ライブまたはスナップショットの記憶領域、またはデータベース・ルート・ファイルを含むすべてのデバイスについて、同期および非同期の読取りおよび書込み I/O カウントを表示します。

「Device IO Overview」画面には、.ruj、.aij または .ace ファイルが格納されているデバイスは含まれません。

データベースに追加または削除した記憶領域は、自動的に画面表示されます。

次の例に、「Device IO Overview」画面を示します。

```
Node: TRIXIE         Oracle Rdb V7.0-00 Performance Monitor 28-MAY-1996 13:07:36
Rate: 3.00 Seconds  Device IO Overview (Unsorted total I/O) Elapsed: 02:46:42.70
Page: 1 of 1     KODD$:[R_ANDERSON.WORK.AIJ]MF_PERSONNEL.RDB;1      Mode: Online
-------------------------------------------------------------------------------
Device.Name.........  Num  Sync.Reads Sync.Writes   Async.Reads Async.Writes
$111$DUA12:            2          52           4             0            0
$111$DUA155:           4           0           0             0            0
$111$DUA46:           19         218           5           144            0
-------------------------------------------------------------------------------
Config Exit Help Menu >next_page <prev_page Options Set_rate Write !
```

"Device.Name" には、展開した記憶領域デバイス名が表示されます。エントリが重複しないように、隠された論理名を展開します。

"Num" には、デバイス情報に含まれるライブおよびスナップショットの記憶領域（ルート・ファイルも含む）の数が表示されます。

その他の欄には、選択した構成に基づいて情報が表示されます。「Device IO Overview」画面のヘッダー領域には、画面の名前が表示されるので、選択した表示構成がわかります。

「Device IO Overview」画面の構成オプションについては、Performance Monitor のヘルプを参照してください。◆

OpenVMS VAX OpenVMS Alpha

「Device Information」画面は、記憶領域デバイスがディスク領域上で減少した場合を判断できます。次に、「Device Information」画面の例を示します。

```
Node: TRIXIE          Oracle Rdb V7.0-00 Performance Monitor 28-MAY-1996 13:10:46
Rate: 3.00 Seconds              Device Information           Elapsed: 02:49:52.38
Page: 1 of 1     USER$:[ORACLEUSER.WORK.AIJ]MF_PERSONNEL.RDB;1    Mode:   Online
------------------------------------------------------------------------------
Device.Name..... Status. #Err Volume.Label FreeBlocks Max#Blocks %Full
$111$DUA46:          Mounted    0 KODA_USER2       377727    2376153  84.1
$111$DUA12:          Mounted    0 KODA_TEST1       595920    1216665  51.0
$111$DUA13:          Mounted    0 KODA_TEST2         6350    1216665  99.4
------------------------------------------------------------------------------
Exit Help Menu >next_page <prev_page Options Set_rate Write !
```

「Device Information」画面は、特定のデータベースにローカルな記憶領域デバイス情報をオンライン表示します。

画面の詳細は、Performance Monitor のヘルプを参照してください。◆

4.2.2 ページ・サイズ

ページ・サイズは、Oracle Rdb がデータベースのページの格納に使用するブロック数であり、1 ブロックは 512 バイトです。デフォルト値は、1 ページあたり 2 ブロックであり、合計で 1024 バイトとなります。

950 ブロックを超える行が存在する場合や、950 バイトを超えるセグメント化文字列が存在する場合には、ページ・サイズを大きくする必要があります。また、Bill of Materials（BOM）アプリケーション（再帰的関係）を実行する場合にも、ページ・サイズを大きくする必要があります。

データベース・ページには、ユーザー・データや空き領域の他に、次のような内容が格納されています。

- ページ・ヘッダー
  固定長であり、使用可能なバイト数など、ページに関する情報が含まれています。
- ライン・インデックス
  ページ上にあるすべてのストレージ・セグメントのディレクトリです。

- レコード・バージョン・ナンバー
  Oracle Rdb では、リレーショナル機能により、レコード型を動的に変更できます。ユーザーは、レコード型の現在の定義に基づいた内容を参照できます。レコード型のバージョンを追跡できるように、各ユーザー・レコードにはバージョン・ナンバー・フィールドがあります。

したがって、ユーザー・データとして使用可能なバイト数は、PAGE SIZE パラメータで指定した実際のページ・サイズよりも小さくなります。このオーバーヘッドは、ページあたり 7 ～ 10% 程度を占めます。オーバーヘッドの大きさは、データベース・ページのサイズによって変わります。ページが大きくなるほど、Oracle Rdb がページ情報の格納に必要とするオーバーヘッドは少なくなります。したがって、データベースの行サイズを計算するときは、最大長の行とオーバーヘッドを考慮して、データベース・ページ・サイズに十分な領域を確保してください。

データの取出しを最も効率的に実行できるのは、データベース・ページがほぼいっぱいの状態のときです。このため、テーブル内の行サイズに合ったページ・サイズを選択してください。ページ・サイズが大きすぎると、領域が無駄になります。順次 I/O は、ページ・サイズが大きい方が効率的ですが、ランダム I/O は、ページ・サイズが小さい方が効率的です。ただし、ページ・サイズが小さすぎると、Oracle Rdb は、複数のデータベースに行を分割（断片化）することがあります。

データベース・ページには、1 行全体を格納できる領域がなくなるまで、空き領域にデータを格納できます。行を変更してサイズを大きくすると、Oracel Rdb は、行を断片化することがあります。行の断片は、使用可能な空き領域に分散されます。行の断片にはポインタがあり、他の断片の位置を示します。ただし、断片化した行を取り出すには、完全な 1 行を取り出すときよりも、多くの時間やリソースを消費します。したがって、データベースの設計や保守では、行が断片化しないように注意してください。

RMU Dump コマンドは、SPAM ページや各データ・ページのダンプを表示できるので、ストレージの割当てやレコードの配置に関する問題の特定に利用できます。Spams_Only 修飾子を指定して RMU Dump コマンドを実行すれば、選択した記憶領域とページ範囲にある SPAM ページのみをダンプできます。RMU Dump コマンドの詳細は、『Oracle RMU Reference Manual』を参照してください。

4.2.3 割当てサイズ

ALLOCATION パラメータで最初のデータベース領域を指定すると、Oracle Rdb は、SQL の CREATE STORAGE AREA 文で定義した各記憶領域について、SQL の CREATE DATABASE 文で指定した PAGE SIZE の値に基づいて、データベース・ファイルにディスク領域を予約します。割当てサイズは、各記憶領域を作成した時点で、行またはレコードの格納に使用される領域の大きさです。ページは、3 ページ単位で割り当てるので、3 ページごとに切り上げます。たとえば、25 ページを割り当てる場合、実際に割り当てられる記憶領域は 27 ページになります。ALLOCATION のデフォルト値は、400 ページです。デフォルトのページ・サイズを使用する場合、記憶領域ファイルの割当てサイズの初期値は、512 バイ

トのディスク・ブロック 800 個です。各記憶領域がどの程度増大するかがわかっている場合や、SQL の EXPORT 文を使用している場合は、各記憶領域の割当てサイズを最終的なサイズに設定します。このように設定すれば、Oracle Rdb は、可能であれば連続した領域を割り当てるので、大規模なデータベース環境で高いシステム・パフォーマンスを発揮します。

4.2.4 ページ・フォーマット

記憶領域のページ・フォーマットのデフォルトは、均一ページ・フォーマットです。完全一致の取出しを行う親子関係などの特殊な場合を除いて、均一ページ・フォーマットはあらゆる環境に適しています。したがって、特殊な環境でなければ、均一ページ・フォーマットを使用してください。均一ページ・フォーマットは、使用するテーブルの数が 10 ～ 15 であり、小～中サイズのシングル・ファイル・データベースを扱うアプリケーションに適しています。また、マルチファイル・データベースでも、一部のテーブルをデフォルトの RDB$SYSTEM 記憶領域内に格納し、別の記憶領域内にもテーブルを個々に格納している場合は、均一ページ・フォーマットが適しています。このようなアプリケーションは、中サイズのテーブルの行にアクセスするときにはソート・インデックスを使用し、小サイズのテーブルの行には、順次アクセスを実行します。このようなアプリケーションで、サイズの大きなテーブルが 1 つまたは複数存在する場合には、複合ページ・フォーマットの記憶領域にハッシュ・インデックスを定義しておけば、行へのアクセスが向上します。

テーブル数が 15 を超える中～大規模の複雑なデータベースを扱うアプリケーションでも、完全一致の取出しを行う親子関係などの特殊な場合を除いて、均一ページ・フォーマットが適しています。大規模で複雑なデータベースでは、複合ページ・フォーマットを慎重に使用すれば効果を発揮できますが、データベース・サイズ、複雑さ、記憶領域タイプを簡単に関連付けることはできません。複合ページ・フォーマットには、ハッシュ・インデックスの使用、SPAM 間隔の最適化、SPAM しきい値の制御、同じ複合記憶領域内での親子行の格納など、いくつかの利点があります。SPAM しきい値の選択と SPAM 間隔の最適化による利点は 4.2.5 項（4-142）と 4.2.6 項（4-146）、親子テーブルでハッシュ・インデックスを使用するときの利点は 3.9.7 項（3-121）を参照してください。

4.2.5 SPAM しきい値の選択の一般的なガイドライン

SPAM ページは、各データ・ページに 2 ビットのエントリを使用して、データ・ページ範囲の空き領域を管理します。2 ビット・エントリは、特定のページのしきい値です。このしきい値は、複合ページ・フォーマットを使用した記憶領域のページや、均一ページ・フォーマットを使用した論理領域のページについて、使用済領域を相対的に示すしきい値を 3 種類設定できます。

しきい値は、ページ上にある空き領域の最大サイズを基準に計算します。ページ上にある空き領域の最大サイズは、ページ・サイズからページ・オーバーヘッドを差し引いて計算します。たとえば、mf_personnel データベース内の DEPARTMENTS 記憶領域では、1 ページ 1024 バイト、オーバーヘッドが 60 バイトだとすると、空き領域の最大サイズは 964 バイトとなります。

SPAM ページとしきい値の詳細は、『Oracle Rdb7 Guide to Database Maintenance』を参照してください。

4.2.5.1 複合ページ・フォーマットでのしきい値

複合ページ・フォーマットを使用した記憶領域を持つマルチファイル・データベースでは、SPAM ページの空き領域インベントリ・リストに対応付けて、しきい値を 3 種類設定できます。デフォルト値は、70%、85%、95% です。これは、各データ・ページ上で保証される空き領域が、それぞれ 30%、15%、5% であることを示します。

テーブルに、インデックスを使用した配置を指定するストレージ・マップがあり、目的のページがいっぱいである場合、バッファ内のページに空き領域がないか探します。バッファ内のページに十分な空き領域がある場合、レコードはそこに格納されます。バッファ内のページに十分な領域がない場合、Oracle Rdb は、記憶領域の SPAM ページをスキャンして、十分な領域がないか探します。SPAM ページをスキャンする前に、バッファ内のページで空き領域を探すことにより、I/O 操作の数が減り、レコードを格納する際のパフォーマンスが向上します。

Oracle Rdb は、SPAM しきい値と配置インデックスを使用して、空き領域を保証するストレージ・アルゴリズムを使用します。Oracle Rdb は、このアルゴリズムでは、3 番目のしきい値に達したページには、新しいレコードを格納しません。したがって、テーブルで PLACEMENT VIA INDEX 句とハッシュ・インデックスを使用する場合、目的のページにすぐに書き込んでしまうことはなく、3 番目のしきい値に達していないことを確認した後で、書込みを行います。

領域に格納する行の長さと、各データベース・ページのサイズがわかれば、しきい値に最適な値を設定し、行の格納を高速化できます。しきい値を設定する目的は、1 つのデータベース・ページに行全体を格納するときに実行する検索処理を最小限に抑えることにあります。しきい値は、データベースに格納されている行のサイズと格納の頻度を基準に設定してください。次に、SPAM しきい値の選択での一般的なガイドラインを示します。

- しきい値には、代表的な行（サイズと格納の頻度において）を、1 つのデータベース・ページ上に 1 回以上格納できるように、保証する値を指定します。
- 記憶領域によって行の長さが大きく変わる場合は、THRESHOLDS ARE オプションを使用して、各記憶領域にしきい値を設定します。
- 各記憶領域に、サイズの異なる複数のテーブル（EMPLOYEES や JOB_HISTORY など）を格納している場合（一般的によく発生する状況）、次の手順に従ってください。
    - 最小のしきい値を設定します。このしきい値は、ページの使用済領域がこのパーセンテージに達していない限り、最大長のレコード型を 1 回以上格納できることを保証します。この設定に対応して、1 ページあたりのブロック数を増やす必要があることがあります。1 ページあたりのブロック数を増やすには、SQL の CREATE DATABASE または SQL の IMPORT 文の PAGE SIZE IS ページ・ブロック BLOCKS を使用してください。

- 2 番目のしきい値を設定します。このしきい値は、ページの使用済領域がこのパーセンテージに達していない限り、最大長よりは小さなレコード型（インデックスまたはテーブル行）を 1 回以上格納できることを保証します。
- Oracle Rdb は、最大のしきい値に達したページには、行を格納しません。このしきい値は、更新によって行が長くなった場合を考えて、ページ上で空き領域を予約するために設定します。

3 番目のしきい値を設定します。データベース・ページがこのしきい値に達すると、完全に使用済とみなされます。3 番目のしきい値を 85% に設定すると、残りの空き領域が 15% 未満のデータベース・ページには、行を格納しません。最も重要な点は、Oracle Rdb は、3 番目のしきい値に達したデータベース・ページを検索しないので、時間を節約できるということです。

注意：SPAM アルゴリズムでは、データ圧縮による行サイズの短縮を前提としていません。したがって、最大長の行サイズには、非圧縮のサイズを使用してください。

各記憶領域のしきい値のバイト数は、次の手順で決定できます。

```
SPAM_THRESHOLD_n = THRESHOLD_n × (maximum free space - 1) /100
```

例 4-24（4-144）は、DEPARTMENTS 記憶領域でデフォルトのしきい値 70%、85%、95% を使用した場合の、各しきい値のバイト数を示しています。

例 4-24 DEPARTMENTS 記憶領域のしきい値の決定

```
SPAM_THRESHOLD_1 = 70 × 963 / 100 = 674
SPAM_THRESHOLD_2 = 85 × 963 / 100 = 818
SPAM_THRESHOLD_1 = 95 × 963 / 100 = 914
```

4.2.5.2 均一ページ・フォーマットでのしきい値

ストレージ・マップやインデックスを作成または変更するとき、論理領域に 3 種類のしきい値を指定できます。データ圧縮やインデックス・ノード・サイズを考慮してしきい値を設定することにより、1 ページに格納可能な行数を最大化できます。Oracle Rdb は、圧縮した行やインデックス・ノード・サイズに対応しています。前の例で、3 番目のしきい値を 94% に設定すると、使用済しきい値は 893 バイト（950 × 0.94）となり、空き領域は 57 バイト残ります。このしきい値は、この論理領域のページに、60 バイトの行を最大限格納することを保証します。SPAM ページのしくみは、『Oracle Rdb7 Guide to Database Maintenance』を参照してください。

論理領域のしきい値の設定は、SQL の CREATE または ALTER INDEX 文、CREATE または ALTER STORAGE MAP 文を使用してください。

次の例は、論理領域にしきい値を設定する方法を示しています。SQL 文の詳細は、『Oracle Rdb7 SQL Reference Manual』を参照してください。

```
SQL> CREATE INDEX index_name
  1>   ON table_name (column_name)
  2>   TYPE IS SORTED
  3>   STORE USING (column_name)
  4>      IN area_name (THRESHOLDS ARE (value1,value2,value3))
  5>      WITH LIMIT OF (literal)
  6>      IN area_name (THRESHOLDS ARE (value1,value2,value3))
  7>      WITH LIMIT OF (literal)
  8>      OTHERWISE IN area_name (THRESHOLDS ARE (value1,value2,value3));
```

次の例では、ソート・インデックスを定義し、STORE 句を使用してインデックスを 3 つの記憶領域に分割しています。THRESHOLDS ARE 句では、各記憶領域に 3 つまでのしきい値を指定できます。

```
SQL> CREATE STORAGE MAP map_name FOR table_name
  1>   STORE USING (column_name)
  2>      IN area_name (THRESHOLDS ARE (value1,value2,value3))
  3>      WITH LIMIT OF (literal)
  4>      IN area_name (THRESHOLDS ARE (value1,value2,value3))
  5>      WITH LIMIT OF (literal)
  6>      OTHERWISE IN area_name (THRESHOLDS ARE (value1,value2,value3))
  7>   ENABLE COMPRESSION;
```

上記の例では、ストレージ・マップを作成し、3 つに分割した領域を指定して論理領域のしきい値を設定し、圧縮を有効化しています。THRESHOLDS ARE (0,0,0) を指定するか、しきい値を指定しない場合、Oracle Rdb は、デフォルトの処理を実行します。つまり、ページに 1 行以上の格納が可能かどうかによって、使用済かどうかをマークします。

```
SQL> CREATE STORAGE MAP map_name FOR table_name
  1>   STORE IN area_name
  2>   ENABLE COMPRESSION
  3>   THRESHOLDS ARE (value1,value2,value3);
```

前の例では、1 つの領域の 1 つのテーブルからすべての行を格納するストレージ・マップを作成し、論理領域のしきい値を設定しています。

```
SQL> ALTER STORAGE MAP map_name
  1>   STORE IN area_name
  2>   ENABLE COMPRESSION
  3>   THRESHOLDS ARE (value1,value2,value3)
  4>   PLACEMENT VIA INDEX index_name;
```

前の例では、PLACEMENT VIA INDEX 句を使用して、記憶領域に行を格納します。指定したしきい値は、CREATE STORAGE MAP 文で追加された領域のみに適用されます。ALTER

STORAGE MAP 文で追加された領域は、THRESHOLDS ARE 句を明示的に指定しなければ、デフォルトの (0,0,0) を使用します。

論理領域のしきい値を指定するには、データベース・ページ上の空き領域と、ページ上に格納される行の圧縮率を計算する必要があります。ページ・サイズの計算は、『Oracle Rdb7 Guide to Database Design and Definition』、データ圧縮オプションの詳細は、4.3.3 項（4-175）を参照してください。

ストレージ・マップやインデックスの作成で使用したオリジナル・テキストは、SQL SHOW STORAGE MAP および SHOW INDEX 文で表示できます。現在のしきい値については、RMU Dump Larea=RDB$AIP または RMU Extract Item=(Index,Storage) コマンドで表示できます。

4.2.6 SPAM 間隔の最適化

データベースが中サイズ（100Mb または 200,000 ブロック未満）で比較的単純な構造（テーブル数は 10 ～ 15 のみ）であれば、SPAM 間隔をデフォルトの 216 ページのままにしておくことができます。一般的に、大きなデータベース（100Mb ～ 1Gb のサイズ）で、テーブル数が 10 ～ 15 しかない場合、非常に大きな記憶領域（10Mb、20,000 ブロック～ 100Mb、200,000 ブロックの間）が存在する可能性があります。記憶領域のサイズが 10Mb ～ 100Mb に増大することがわかっている場合は、記憶領域パラメータを調整してください。上記の場合は、記憶領域でアプリケーションの処理内容（挿入集中型か新集中型か）に合わせて、SPAM 間隔を調整します。サイズが 10Mb、ページ・サイズが 2 ブロックの記憶領域に、10,000 ページあるとすると、デフォルトの SPAM 間隔は 216 ページなので、SPAM ページはおよそ 46 になります。100Mb の記憶領域には、460 の SPAM ページが存在するので、空き領域を探すための I/O 操作も増大します。

更新集中型の環境では、SPAM 間隔に大きな値を設定すると、SPAM ページの競合が発生しやすくなります。空き領域情報は、多数のページの情報が 1 つの SPAM ページにまとめて格納されているので、あるユーザーが、SPAM ページが管理しているデータ・ページの行を更新しようとすると、他のユーザーが、その SPAM ページの空き領域インベントリ・リストを使い終わるまでの間、待機しなければなりません。

次に、このような INTERVAL IS のトレード・オフの関係をまとめます。

- 記憶領域に大きな値を間隔として設定すると、挿入集中型のアプリケーションでは、Oracle Rdb が空き領域を探すときのディスク I/O を減らすことができます。ただし、大きな値を設定すると、更新集中型アプリケーションでは、SPAM ページが管理する記憶領域に多数のユーザーが同時にアクセスするため、SPAM ページのロックが増大します。
- 間隔を小さくすると、更新集中型のアプリケーションでも、SPAM ページのロックを減少できます。ただし、挿入集中型のアプリケーションでは、Oracle Rdb が空き領域を探すときのディスク I/O（および経過時間）が増大します。

ユーザーは、各アプリケーション環境では、ディスク I/O と SPAM ページのロックのどちらを優先するかを検討し、INTERVAL IS の値を調整してください。ディスク I/O 操作の数は、Performance Monitor の「IO Statistics(by file)」画面で参照できます。Performance Monitor の画面の起動と説明は、2.2 項（2-15）を参照してください。

間隔やしきい値の設定に関する構文と説明は、『Oracle Rdb7 SQL Reference Manual』を参照してください。間隔やしきい値の設定例や、しきい値の意味やアプリケーションでの使用方法など、SPAM しきい値の詳細な説明は、『Oracle Rdb7 Guide to Database Maintenance』を参照してください。

4.2.7 異なるディスクへのスナップショット、記憶領域、データベース・ファイルの配置

シングル・ファイル・データベースでは、スナップショット・ファイル（.snp）やデータベース・ルート・ファイル（.rdb）を別のディスク・デバイスに配置すると、ディスク I/O の競合が大幅に低減することがあります。マルチファイル・データベースでは、記憶領域ファイル（.rda）を、.rdb や .snp とは別のディスクに移動することにより、ディスク I/O の競合を軽減できます。.ruj および .aij ファイルの配置によって、ディスクの競合を最小限に軽減でき、データベース・リカバリも効率的に実行できます。.aij ファイルは、他のデータベース・ファイルと同じディスクに格納しないでください。

読取り専用トランザクションや読取り / 書込みトランザクションが同時に発生するアプリケーションでは、同じ記憶領域にある .snp ファイルと .rda ファイルを別に配置すると、パフォーマンスが向上します。また、読取り / 書込みトランザクションが大量に発生するアプリケーションでは、スナップショットを有効にし、デフォルトの IMMEDIATE オプションを使うと、I/O 操作を 2 つのディスクに分散できるので、パフォーマンスが向上します。また、ディスクへの書込みは、コミット時に非同期的に実行されるので、I/O 操作を複数のディスクに分散することによって、応答時間が向上します。

各 .rda ファイルの .snp ファイルの位置は、SQL の CREATE DATABASE 文の CREATE STORAGE AREA 句で指定します。.snp ファイルの名前は、ALTER DATABASE 文の ALTER STORAGE AREA 句では変更できません。

4.2.8 スナップショット・ファイルの割当ての初期化、移動および変更

.snp ファイルの割当ての初期化、移動、変更は、RMU Repair コマンドで実行できます。このような操作は、スナップショット・ファイルを格納しているディスクにハードウェア障害が発生したとき、ハードウェアのアップグレードのために取り外すとき、スナップショット・ファイルが大きくなりすぎて切り捨てるときなどに必要です。オラクル社は、RMU Repair コマンドを実行する前に、RMU Backup コマンドを実行して、データベース全体をバックアップすることをお薦めします。

Area 修飾子を指定して RMU Repair Nospams Initialize=Snapshots コマンドを実行すると、特定の記憶領域のスナップショット・ファイルを初期化できます。例 4-25（4-148）は、mf_personnel データベースの departments.snp と jobs.snp を初期化する方法を示しています。

例 4-25 RMU Repair を実行し、データベース内の特定の記憶領域の既存のスナップショット・ファイルを初期化

```
$ RMU/REPAIR/NOSPAMS/INITIALIZE=SNAPSHOTS/AREA=(DEPARTMENTS,JOBS) mf_personnel
$ rmu -repair -nospams -initialize=snapshots
> -area=(DEPARTMENTS,JOBS) mf_personnel
```

例 4-26（4-148）のように、Area 修飾子を指定しないで RMU Repair Nospams Initialize=Snapshots コマンドを実行すると、mf_personnel データベースのすべてのスナップショット・ファイルを初期化できます。

例 4-26 RMU Repair を実行し、データベース内の既存のスナップショット・ファイルをすべて初期化

```
$ RMU/REPAIR/NOSPAMS/INITIALIZE=SNAPSHOTS mf_personnel
$ rmu -repair -nospams -initialize=snapshots mf_personnel
```

Initialize=Snapshots 修飾子と CONFIRM キーワードを指定すると、初期化だけでなく、スナップショットの割当ての名前の変更、移動、変更をオプションで実行できます。CONFIRM キーワードを指定すると、Oracle Rdb は、1 つ以上のスナップショット・ファイルの名前を割当てをプロンプト表示します。Area 修飾子を使うと、データベース内で変更したいスナップショット・ファイルを選択できます。Area 修飾子を省略すると、データベースのすべてのスナップショット・ファイルが初期化され、Oracle Rdb は、各スナップショット・ファイルの別名と割当てを、インタラクティブにプロンプト表示します。スナップショット・ファイルの新しいファイル名を指定すれば、スナップショット・ファイルの位置を変更できます。スナップショット・ファイルの新しい割当てを指定すれば、スナップショット・ファイルのサイズを増減できます。

OpenVMS VAX OpenVMS Alpha

例 4-27（4-148）では、RMU Repair コマンドを使用して、departments.snp の初期化と名前の変更、salary_history.snp の初期化と移動、jobs.snp の初期化、移動および切捨てを行っています。

例 4-27 RMU Repair を実行し、データベース内の特定の記憶領域スナップショット・ファイルを初期化、名前の変更および切捨て

```
$ RMU/REPAIR/NOSPAMS/INITIALIZE=SNAPSHOTS=CONFIRM -
_$ /AREA=(DEPARTMENTS,JOBS,SALARY_HISTORY) mf_personnel
%RMU-I-FULBACREQ, A full backup of this database should be performed after
RMU/REPAIR
Area DEPARTMENTS snapshot filename [SQL1:[TEST]DEPARTMENTS.SNP;1]: NEW_DEPT
Area DEPARTMENTS snapshot file allocation [10]?
Area SALARY_HISTORY snapshot filename [SQL1:[TEST]SALARY_HISTORY.SNP;1]: SQL2:
Area SALARY_HISTORY snapshot file allocation [10]?
Area JOBS snapshot filename [SQL1:[TEST]JOBS.SNP;1]: SQL2:[TEST2]
```

```
Area JOBS snapshot file allocation [10]? 5
$
```

◆

スナップショット・ファイルの新しい位置と割当ては、Performance Monitor の「Database Parameter Information」サブメニューから「Storage Area Information」画面を選択すると、表示できます。

RMU Repair でスナップショット・ファイルの新しい名前を指定するときに、注意が必要です。すでに作成済で存在するファイルの名前を指定すると、指定に従って初期化されてしまいます。

このように、現在使用中のデータベース・ファイルを誤って初期化してしまった場合は、エラー修正が完了するまでは、データベースを使わないでください。RMU Restore コマンドを実行すると、RMU Repair コマンドの前に実行したバックアップの状態にまで、データベースを戻すことができます。RMU Repair コマンドの前にデータベースをバックアップしていない場合は、最も新しいバックアップ・ファイルを使用してリストアし、.aij ファイルからのリカバリを実行してください（ただし、After-image ジャーナルが有効になっている場合）。

スナップショット・ファイルに誤った名前を指定した場合（たとえば、例 4-27（4-148）のすべてのスナップショット・ファイルに JOBS.SNP と指定）、正しい名前を指定し、RMU Repair コマンドを再実行してください。

RMU Repair コマンドが完了したら、古いスナップショット・ファイルを削除し、RMU Backup コマンドを実行して、データベース全体をバックアップします。

スナップショット・ファイルは、オンラインで切り捨てることができます。したがって、スナップショット・ファイルを切り捨てている間も、ユーザーはデータベース・ファイルを使用できます。

例 4-28（4-149）は、データベースまたは記憶領域のスナップショット・ファイルをオンラインで切り捨てる方法を示しています。

例 4-28 データベースまたは記憶領域のスナップショット・ファイルのオンラインでの切捨て

```
SQL> -- Truncate the database snapshot file on line:
SQL> ALTER DATABASE FILENAME mf_personnel
  1>   SNAPSHOT ALLOCATION IS 18 PAGES;
SQL> --
SQL> -- Truncate a storage area snapshot file on line:
SQL> ALTER DATABASE FILENAME mf_personnel
  1>   ALTER STORAGE AREA EMPIDS_LOW
  2>      SNAPSHOT ALLOCATION IS 6 PAGES;
```

オンラインでスナップショット・ファイルの切捨てを開始すると、Oracle Rdb は、読取り専用トランザクションの静止ポイントを待ち、遅延スナップショット・モードを一時的に採用します。遅延スナップショット・モードが有効になると、Oracle Rdb は、スナップショット・ファイルを要求したサイズにまで切り捨てます。切捨て処理の間、読取り専用トランザ

クションはブロックされます。スナップショット・ファイルの切捨てが完了すると、Oracle Rdb は、データベースを前のスナップショット・モードに戻します（スナップショット・ファイルの切捨て処理中は、遅延スナップショット・モードを適用）。

スナップショットを切り捨てる間、ユーザーは、データベースの接続時に新しくロックを 2 つ取得するので、データベースへの接続時間が大幅に増加します。

次のような場合には、スナップショット・ファイルを切り捨てることをお薦めします。

- 長い読取り専用トランザクションの後
- サイズの大きなスナップショット・ファイルの断片化により、パフォーマンスが低下するのを防止

4.2.9 スナップショット・ファイルの増大と事前起動済トランザクション

事前起動済トランザクションは、トランザクションの開始時に、Oracle Rdb が使用する I/O の最適化方法です。事前起動済トランザクションを有効に（デフォルト）しておくと、プロセスが COMMIT 文または ROLLBACK 文を発行して読取り / 書込みトランザクションを終了するときに、Oracle Rdb は、そのプロセスで新しい読取り / 書込みトランザクションをすぐに開始します（新しく開始された読取り / 書込みトランザクションは、**事前起動済トランザクション**と呼ばれます）。Oracle Rdb は、この新しい事前起動済トランザクションに、元の読取り / 書込みトランザクションと同じトランザクション・シーケンス番号（Transaction Sequence Number：TSN）を割り当てます。Oracle Rdb は、事前起動済トランザクションには、データベース・ルート・ファイルから新しい TSN を予約する必要がないので、I/O を軽減できます。

事前起動済トランザクションは、無効にもできます。事前起動済トランザクションによって、データベースのスナップショット・ファイルのサイズが大きくなりすぎる場合には、事前起動済トランザクションを無効にできます。事前起動済トランザクションを無効にすると、新しいトランザクションによって発生するルート・ファイルへの I/O は増加します。

図 4-11（4-151）は、事前起動済トランザクションを有効にした場合に、Oracle Rdb が、スナップショット・レコードの書込みを処理する手順を示しています。図 4-11（4-151）では、4 つのトランザクションがデータベースにアクセスしています。最初のトランザクションは読取り / 書込みトランザクション、2 番目は読取り専用トランザクション、3 番目は読取り / 書込みトランザクション、4 番目は読取り専用トランザクションです。Oracle Rdb は、データベース内で最も古い読取り / 書込みトランザクションに、" カットオフ TSN" を割り当てます。スナップショット・ページ上のレコードの TSN がカットオフ TSN よりも小さい場合、Oracle Rdb は、レコードを再要求できます。

図 4-11 事前起動済トランザクションとスナップショット・ファイルの増大

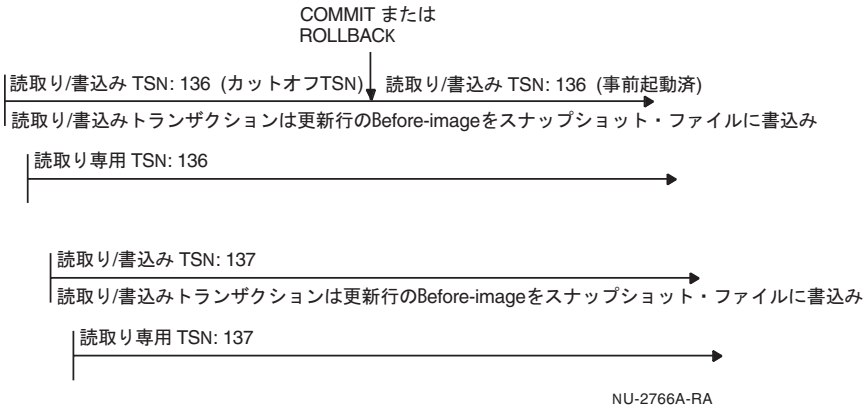


図 4-11（4-151）では、事前起動済トランザクションが有効になっているため、最初の読取り / 書込みトランザクション（TSN 136、カットオフ TSN）をコミットまたはロールバックするときに、新しい読取り / 書込みトランザクションが事前開始します。Oracle Rdb は、事前起動済トランザクションに TSN 136 を割り当てるので、TSN 136 はカットオフ TSN のままです。最初の読取り / 書込みトランザクション（TSN 136）は、データベース内で最も古い読取り / 書込みトランザクションなので、Oracle Rdb は、カットオフ TSN のプロセスと他の読取り / 書込みトランザクション（TSN 137）で書き込んだスナップショット・レコードを再要求できません。次のいずれかの条件が満たされるまで、スナップショット・ファイルにはレコードが追加されます。

- カットオフ TSN のプロセスが、データベースから切断または終了した場合。
- カットオフ TSN のプロセスが、コミットまたはロールバックを実行し、即時にロールバックする読取り専用トランザクションが明示的に開始した場合。

```
SQL> COMMIT;
SQL> SET TRANSACTION READ ONLY;
SQL> ROLLBACK;
```

通常は、事前起動済トランザクションを有効にしても、スナップショット・ファイルが大きくなりすぎることはありません。ただし、アプリケーションが使用しているサーバーが、長時間データベースに接続している場合、カットオフ TSN は、非常に古くなっている可能性があります。このような場合、Oracle Rdb が再要求できるスナップショット・レコードは少なくなるので、スナップショット・ファイルのサイズは大きくなっていることがあります。

例 4-29（4-152）では、RMU Dump Users コマンドの出力を使用して、図 4-11（4-151）で示した事前起動済トランザクションをさらに詳細に説明します。例 4-29（4-152）では、図 4-

11（4-151）で示した 2 つの読取り / 書込みトランザクションと、2 つの読取り専用トランザクションを使用します。事前起動済トランザクションは、読取り / 書込みトランザクションに使用します。最初の読取り / 書込みトランザクション（TSN 136、カットオフ TSN）をコミットまたはロールバックした後、Oracle Rdb は、同じ TSN で事前起動済トランザクションを開始するので、このトランザクションがカットオフ TSN のままになります。

例 4-29 事前起動済トランザクションを有効にした場合のトランザクション

```
$! While the four transactions are in progress:
$ RMU/DUMP/USERS mf_personnel
Active user with process ID 71A01142
    Stream ID is 1
    Monitor ID is 1
    Transaction ID is 19
    Recovery journal filename is "SQL_USER1:[RDM$RUJ]MF_PERSONNEL$009685067AF25
    C00.RUJ;1"
    Read/write transaction in progress (1)
    Transaction sequence number is 136 (2)
Active user with process ID 71A0126C
    Stream ID is 1
    Monitor ID is 1
    Transaction ID is 86
    Snapshot transaction in progress
    Transaction sequence number is 136 (3)
Active user with process ID 71A01269
    Stream ID is 1
    Monitor ID is 1
    Transaction ID is 166
    Recovery journal filename is "SQL_USER1:[RDM$RUJ]MF_PERSONNEL$0096850694F0
    B7A0.RUJ;1"
    Read/write transaction in progress
    Transaction sequence number is 137 (4)
Active user with process ID 71A0126D
    Stream ID is 1
    Monitor ID is 1
    Transaction ID is 246
    Snapshot transaction in progress
    Transaction sequence number is 136 (3)
$!
   .
   .
   .
$! After the first read/write transaction (the cutoff TSN) has
$! committed or rolled back:
$ RMU/DUMP/USERS mf_personnel
Active user with process ID 71A01142
    Stream ID is 1
```

```
    Monitor ID is 1
    Transaction ID is 19
    Recovery journal filename is "SQL_USER1:[RDM$RUJ]MF_PERSONNEL$009685067AF2
    5C00.RUJ;1"
    Read/write transaction in progress (5)
    Transaction sequence number is 136 (6)
Active user with process ID 71A0126C
    Stream ID is 1
    Monitor ID is 1
    Transaction ID is 86
    Snapshot transaction in progress
    Transaction sequence number is 136 (7)
Active user with process ID 71A01269
    Stream ID is 1
    Monitor ID is 1
    Transaction ID is 166
    Recovery journal filename is "SQL_USER1:[RDM$RUJ]MF_PERSONNEL$0096850694F0
    B7A0.RUJ;1"
    Read/write transaction in progress
    Transaction sequence number is 137
Active user with process ID 71A0126D
    Stream ID is 1
    Monitor ID is 1
    Transaction ID is 246
    Snapshot transaction in progress
    Transaction sequence number is 136 (7)
$
```

次の番号は、例 4-29（4-152）で示したパラメータの説明です。

(1) 最初の読取り / 書込みトランザクションは、データベースの中で最も古い読取り / 書込みトランザクションなので、これがカットオフ TSN になります。

(2) カットオフ TSN は TSN 136 です。

(3) 読取り専用（スナップショット）トランザクションには、最も古い読取り / 書込みトランザクション（カットオフ TSN）の TSN が割り当てられます。

(4) 2 番目の読取り / 書込みトランザクションには、次の TSN である 137 が割り当てられます。

(5) カットオフ TSN をコミットまたはロールバックすると、Oracle Rdb は、新しい読取り / 書込みトランザクションを自動的に事前起動します。

(6) 事前起動済トランザクションの TSN は、オリジナルのトランザクションと同じであるため、このトランザクションは、カットオフ TSN のままです。

(7) 最も古い読取り / 書込みトランザクションは TSN 136 なので、Oracle Rdb は、カットオフ TSN のプロセスと他の読取り / 書込みトランザクション（TSN 137）で書き込んだスナップショット・レコードを再要求できません。

例 4-30（4-154）は、事前起動済トランザクションを無効にした場合、図 4-11（4-151）で示したトランザクションにどのような影響を与えるかを示しています。例 4-30（4-154）では、図 4-11（4-151）で示した 2 つの読取り / 書込みトランザクションと、2 つの読取り専用トランザクションを使用します。ただし、例 4-29（4-152）とは異なり、ここでは事前起動済トランザクションを無効にします。最初の読取り / 書込みトランザクション（TSN 136、カットオフ TSN）をコミットまたはロールバックした後、Oracle Rdb は、事前起動済トランザクションを開始しないので、RMU Dump Users には、"No transaction in progress" というメッセージが出力されます。これにより、Oracle Rdb は、カットオフ TSN のコミットまたはロールバックの前に書き込まれたスナップショット・レコード（TSN が 136 以下のすべてのスナップショット・レコード）を再要求できます。

例 4-30 事前起動済トランザクションを無効にした場合のトランザクション

```
$! While the four transactions are in progress:
$ RMU/DUMP/USERS mf_personnel
Active user with process ID 71A01142
    Stream ID is 1
    Monitor ID is 1
    Transaction ID is 19
    Recovery journal filename is "SQL_USER1:[RDM$RUJ]MF_PERSONNEL$009685067AF2
    5C00.RUJ;1"
    Read/write transaction in progress (1)
    Transaction sequence number is 136 (2)
Active user with process ID 71A0126C
    Stream ID is 1
    Monitor ID is 1
    Transaction ID is 86
    Snapshot transaction in progress
    Transaction sequence number is 136 (3)
Active user with process ID 71A01269
    Stream ID is 1
    Monitor ID is 1
    Transaction ID is 166
    Recovery journal filename is "SQL_USER1:[RDM$RUJ]MF_PERSONNEL$0096850694F0
    B7A0.RUJ;1"
    Read/write transaction in progress
    Transaction sequence number is 137 (4)
Active user with process ID 71A0126D
    Stream ID is 1
    Monitor ID is 1
    Transaction ID is 246
    Snapshot transaction in progress
```

```
    Transaction sequence number is 136 (3)
$!
   .
   .
   .
$! After the first read/write transaction (the cutoff TSN) has
$! committed or rolled back:
$ RMU/DUMP/USERS mf_personnel
Active user with process ID 71A01142
    Stream ID is 1
    Monitor ID is 1
    Transaction ID is 19
    Recovery journal filename is "SQL_USER1:[RDM$RUJ]MF_PERSONNEL$009685067AF2
    5C00.RUJ;1"
    No transaction in progress (5)
Active user with process ID 71A0126C
    Stream ID is 1
    Monitor ID is 1
    Transaction ID is 86
    Snapshot transaction in progress
    Transaction sequence number is 136 (7)
Active user with process ID 71A01269
    Stream ID is 1
    Monitor ID is 1
    Transaction ID is 166
    Recovery journal filename is "SQL_USER1:[RDM$RUJ]MF_PERSONNEL$0096850694F0
    B7A0.RUJ;1"
    Read/write transaction in progress
    Transaction sequence number is 137 (6)
Active user with process ID 71A0126D
    Stream ID is 1
    Monitor ID is 1
    Transaction ID is 246
    Snapshot transaction in progress
    Transaction sequence number is 136 (7)
$
```

次の番号は、例 4-30（4-154）で示したパラメータの説明です。

(1) 最初の読取り / 書込みトランザクションは、データベースの中で最も古い読取り / 書込みトランザクションなので、これがカットオフ TSN になります。

(2) カットオフ TSN は TSN 136 です。

(3) 読取り専用（スナップショット）トランザクションには、最も古い読取り / 書込みトランザクション（カットオフ TSN）の TSN が割り当てられます。

(4) 2 番目の読取り / 書込みトランザクションには、次の TSN である 137 が割り当てられます。

(5) カットオフ TSN をコミットまたはロールバックすると、Oracle Rdb は、新しいトランザクションの事前開始は行いません。"No transaction in progress" が表示されます。

(6) 最初の読取り / 書込みトランザクションは、コミットまたはロールバックで終了し、2 番目の読取り / 書込みトランザクション（TSN 137）が、データベースの中で最も古い読取り / 書込みトランザクションとなります（カットオフ TSN）。

(7) 最も古い読取り / 書込みトランザクションは TSN 137 なので、Oracle Rdb は、TSN が 136 以下のスナップショット・レコードを再要求できます。ここでは、Oracle Rdb は、TSN 136 のプロセスで書き込んだスナップショット・レコードを再要求します。

事前起動済トランザクションの有効化は、SQL の ATTACH、CONNECT、DECLARE ALIAS、CREATE DATABASE、IMPORT 文で PRESTARTED TRANSACTIONS ARE ON 句を指定または省略します。

事前起動済トランザクションの無効化は、SQL の ATTACH、CONNECT、DECLARE ALIAS、CREATE DATABASE、IMPORT 文で PRESTARTED TRANSACTIONS ARE OFF 句を指定します。

CREATE DATABASE または IMPORT 文で PRESTARTED TRANSACTIONS ARE ON または PRESTARTED TRANSACTIONS ARE OFF 句を指定すると、現在の接続のみが対象になります。したがって、永続的なデータベース属性にはなりません。

SQL の ATTACH、CONNECT、DECLARE ALIAS、CREATE DATABASE、IMPORT 文での PRESTARTED TRANSACTIONS ARE ON と PRESTARTED TRANSACTIONS ARE OFF 句の詳細は、『Oracle Rdb7 SQL Reference Manual』を参照してください。

4.3 ストレージ・マップ・パラメータの調整

この項では、RMU Analyze Placement コマンドと、2 つのストレージ・マップ・パラメータ（PLACEMENT VIA INDEX とデータ圧縮）の詳細と、データベース・アプリケーションでの最適値を選択する方法を説明します。Oracle Rdb では、この 2 つのストレージ・マップ・パラメータをデフォルト値に設定しても十分なパフォーマンスを発揮できるので、特殊なアプリケーション向けにデータベースを調整するときのガイドラインとして活用してください。

デフォルト設定があるのは、2 つのストレージ・マップ・パラメータのみであり、圧縮は有効化、論理領域しきい値は無効化されています。その他のストレージ・マップ・パラメータは、明示的に宣言してください（マップ名を入力するなど）。

表 4-11（4-157）は、SQL 文の説明であり、SQL の CREATE DATABASE、ALTER DATABASE、IMPORT 文で、ストレージ・マップ・パラメータの指定が可能かどうかをまとめています。

表 4-11 SQL 文でのストレージ・マップ・パラメータの指定

ストレージ・マップ・パラメータ	SQL CREATE STORAGE MAP	SQL ALTER STORAGE MAP	SQL IMPORT または CREATE STORAGE MAP
マップ名	可	可	可
説明	不可	不可	不可
テーブルの指定	可	不可	可
行の使用	可	可	可
記憶領域名内の指定	可	可	可
分割リミットの指定	可	可	可
PLACEMENT VIA INDEX の指定	可	可	可
圧縮の有効化 / 無効化の指定	可	可	可
論理値はしきい値の指定	可	可	可
REORGANIZE の指定	不可	可	不可

また、CREATE または ALTER INDEX 文、CREATE または ALTER INDEX 文を使えば、均一ページ・フォーマット内での論理領域のページしきい値を設定できます。しきい値の設定の詳細は、4.2.5 項（4-142）を参照してください。

ストレージ・マップ・パラメータと、異なる記憶領域のデータの再編成は、『Oracle Rdb7 Guide to Database Design and Definition』を参照してください。

4.3.1 ストレージ・マップ・パラメータ情報の収集

この項では、RMU Analyze コマンドを使用し、Placement および Option [= Normal, Full, Debug のいずれか] 修飾子を指定した場合の出力フォーマットと内容について説明します。RMU Analyz コマンドの使用方法に関する一般的な内容は、2.1 項（2-1）を参照してください。

RMU Analyze Placement コマンドは、次の情報を収集するときに便利です。

- 最大と平均のパスの長さ、つまり、データ行にアクセスするのに必要なインデックス・レコードの数の最大値と平均値。
- 最大 I/O パスの長さ、つまり、データ行に達するまでに横断するページの合計数最大 I/O パスの長さとは、次の条件がすべて満たされたときの I/O。
  - 要求されたデータベース・レコードがバッファ内に存在しない（ローカルまたはグローバル・バッファ）。

    - 使用可能なグローバル・バッファの数は最小限。
    - データベース・レコードを要求する他のユーザー間で、非常に大きな競合が発生。
- 最小 I/O パスの長さ、またはインデックスとデータ行がバッファ内に存在するかどうかバッファ・サイズを考慮する。最小 I/O パスの長さとは、次の条件がすべて満たされたときの I/O。
    - 要求されたデータベース・レコードがバッファ内に存在しない（ローカルまたはグローバル・バッファ）。
    - 十分な数のバッファがあり、I/O を軽減。
    - データベース・レコードを要求する他のユーザーによる競合はない。
- 指定したインデックスでの、DBkey パスの長さの頻度分布、最大 I/O パスの長さ、最小 I/O パスの長さ。
- 論理領域 ID と DBkey ごとに示した記憶領域内のデータ・ページ上のデータ行の分布、各データ行へのアクセスに必要なキーの数、データ行へのアクセスに必要な最大および最小 I/O パスの長さ、各キーの長さ、データ行の各キー。

最大 I/O パスの長さは、最悪の場合を考えて計算します。したがって、実際のアプリケーション・パフォーマンスは、最大 I/O パスの長さよりも低下することはありません。

ただし、最小 I/O パスの長さよりも、実際のアプリケーション・パフォーマンスの方がよい場合もあります。たとえば、1 回目のインデックス・ルックアップでレコードはバッファ内にフェッチされているので、2 回目またはそれに続くインデックス・ルックアップでは、必要なレコードがすべてバッファ内で検索できる場合、I/O はほとんどまたはまったく必要なくなります。本当の最小 I/O パスの長さは、非常に 0 に近い値です。ほとんどの場合、最小 I/O パスの長さの推定値は、アプリケーションで測定した I/O パスの長さになります。

4.3.1.1 RMU Analyze Placement Option=Normal コマンドの使用方法

RMU Analyze Placement コマンドで Option=Normal 修飾子と EMPLOYEES_HASH ハッシュ・インデックスを指定すると、Oracle RMU は、例 4-31（4-158）のような内容を出力します。

例 4-31 RMU Analyze Placement コマンドで、Option=Normal 修飾子とハッシュ・インデックスを指定

```
$ RMU/ANALYZE/PLACEMENT mf_personnel EMPLOYEES_HASH /OPTION=NORMAL
-------------------------------------------------------------------------------
 Indices for database  - $DUA0:[ORION]MF_PERSONNEL.RDB;
-------------------------------------------------------------------------------
 Hashed index EMPLOYEES_HASH for relation EMPLOYEES duplicates not allowed
 Levels: 1, Nodes: 69, Keys: 100, Records: 100
 Maximum path length --  DBkeys: 3, IO range: 1 to 1
```

```
 Average path length --  DBkeys: 3.00, IO range: 1.00 to 1.00
-------------------------------------------------------------------------------
```

次に、RMU Analyze Placement コマンドで Option=Normal 修飾子を指定した場合の出力内容を一覧します。説明の後に続くカッコで囲んだ数字は、例 4-31（4-158）の EMPLOYEES_HASH インデックスに対応しています。

- インデックス名（EMPLOYEES_HASH）
- for relation
  テーブル名（EMPLOYEES）
- duplicates
  重複を許可するかしないか（duplicates not allowed）
- Levels
  インデックス・レベルの数（1）
- Nodes
  インデックス内のノードの合計数（69）
- Keys
  インデックス内の一意キーの合計数（100）
- Records
  インデックス内の一意キーを持つデータ行の合計数（100）
- Maximum path length---DBkeys
  データ行に達するまでにアクセスする DBkey（インデックス・レコード）の最大数（3）
- IO range
  データ行へのアクセスに必要な最大 I/O パスの長さの範囲。最初の値は、データ行へのアクセスに必要な I/O の、低い推定値です。2 番目の値は、データ行へのアクセスに必要な I/O の、高い推定値です（1to1）。
- Average path length---DBkeys
  データ行に達するまでにアクセスする DBkey（インデックス・レコード）の平均（3.00）
- IO range
  データ行へのアクセスに必要な平均 I/O パスの長さの範囲。最初の値は、データ行へのアクセスに必要な I/O の平均値の、低い推定値です。2 番目の値は、データ行へのアクセスに必要な I/O の平均値の、高い推定値です（1.00 to 1.00）。

RMU Analyze Placement コマンドを Option=Normal 修飾子を指定し、DEPARTMENTS_INDEX ソート・インデックスを指定すると、Oracle RMU は、例 4-32（4-160）のような内容を出力します。

例 4-32 RMU Analyze Placement コマンドで、ランクなしソート・インデックスに Option=Normal 修飾子を指定

```
$ RMU/ANALYZE/PLACEMENT mf_personnel DEPARTMENTS_INDEX /OPTION=NORMAL
-------------------------------------------------------------------------------
 Indices for database  - $DUA0:[ORION]MF_PERSONNEL.RDB;
-------------------------------------------------------------------------------
 Index DEPARTMENTS_INDEX for relation DEPARTMENTS duplicates not allowed
 Levels: 1, Nodes: 1, Keys: 26, Records: 26
 Maximum path length --  DBkeys: 2, IO range: 1 to 2
 Average path length --  DBkeys: 2.00, IO range: 1.00 to 1.65
-------------------------------------------------------------------------------
```

例 4-32（4-160）の DEPARTMENTS_INDEX ソート・インデックスでは、最大レベルは 1 なので、ノードは 1 つです（ルート・ノード）。26 のデータ行には、26 の一意キーがあります。最大パスの長さ（アクセスしたレコード数、リーフ・ノードとデータ行）は 2 です。最大パスの長さで、データ行へのアクセスに必要な I/O の推測値は、1（低い推測値）から 2（高い推測値）の範囲です。すべてのインデックス・データを平均したパスの長さ（アクセスしたレコード数）は 2.00、データへのアクセスに必要な I/O の推測値を、すべてのインデックス・データで平均すると、1.00（低い推測値）から 1.65（高い推測値）の範囲になります。

例 4-33（4-160）は、RMU Analyze Placement コマンドで Option=Normal を指定し、重複可能なランク付ソート・インデックスを指定した場合の出力内容を示します。

例 4-33 RMU Analyze Placement コマンドで、ランク付きソート・インデックスに Option=Normal 修飾子を指定

```
$ RMU/ANALYZE/PLACEMENT mf_personnel DEGREES_YEAR_RANKED /OPTION=NORMAL
-------------------------------------------------------------------------------
 Indices for database  - $DUA0:[ORION]MF_PERSONNEL.RDB;
-------------------------------------------------------------------------------
 Index DEGREES_YEAR_RANKED for relation DEGREES duplicates allowed
 Levels: 2, Nodes: 5, Keys: 25, Records: 165
   Dup nodes: 0, Dup keys: 18, Dup maps: 18, Dup records: 162
 Maximum path length --  DBkeys: 3, IO range: 2 to 3
 Average path length --  DBkeys: 3.00, IO range: 2.01 to 2.48
-------------------------------------------------------------------------------
```

インデックスは重複可能なので、RMU Analyze Placement コマンドは、次のような内容を追加で表示します。

- Dup nodes
  ランク付きソート・インデックスに関するオーバーフロー・ノードの数です。インデックスに重複がある場合でも、この値は 0 になる可能性があります（0）。ランクなしソート・インデックスでは、重複ノードの数を示します。ハッシュ・インデックスでは、重複ノードの数を示します。

- Dup keys
  インデックス内の重複キーの合計数です。
- Dup maps
  ランク付きソート・インデックスのみは、重複インデックス・キー・データをポイントする DBkey を表すビット・マップの数です。このフィールドは、ランクなしソート・インデックスやハッシュ・インデックスには表示されません。
- Dup records
  インデックス内の重複レコードの合計数です。

RMU Analyze Placement コマンドで Option=Normal 修飾子を指定して JOB_HISTORY_HASH ハッシュ・インデックスを指定すると、重複が許可されるので、Oracle RMU は、例 4-34（4-161）のような内容を出力します。

例 4-34 RMU Analyze Placement コマンドで、Option=Normal 修飾子とハッシュ・インデックスを指定（重複可）

```
$ RMU/ANALYZE/PLACEMENT mf_personnel JOB_HISTORY_HASH /OPTION=NORMAL
-------------------------------------------------------------------------------
 Indices for database  - $DUA0:[ORION]MF_PERSONNEL.RDB;
-------------------------------------------------------------------------------
 Hashed index JOB_HISTORY_HASH for relation JOB_HISTORY duplicates allowed
 Levels: 1, Nodes: 69, Keys: 100, Records: 274
   Dup nodes: 80, Dup keys: 80, Dup records: 254
 Maximum path length --  DBkeys: 4, IO range: 1 to 3
 Average path length --  DBkeys: 3.93, IO range: 1.00 to 1.08
-------------------------------------------------------------------------------
```

例 4-34（4-161）の JOB_HISTORY_HASH ハッシュ・インデックスでは、レベルが 1、ノード数は 69 です（ハッシュ・バケット）。274 のデータ行には、合計で 100 のキーがあります。キーは 80 あり（重複ノード・レコード）、データ行は 254 です。（したがって、重複しないキーは 20、データ・レコードは 20 です）。

最大パスの長さは 4（アクセスしたレコード数。システム・レコード、ハッシュ・バケット、重複ノード・レコード、データ行）、最大パスの長さでデータ・アクセスに必要な I/O の推定値は、1（低い推測値）から 3（高い推測値）の範囲です。すべてのインデックス・データを平均したパスの長さ（アクセスしたレコード数）は 3.93、データへのアクセスに必要な I/O の推測値を、すべてのインデックス・データで平均すると、1.00（低い推測値）から 1.08（高い推測値）の範囲です。

ソート・インデックスでは、最上位レベル（ルート・ノード）、最下位レベル（リーフ・ノード）およびデータ・ページを読み取る場合、行を取り出すには、I/O 操作を 3 回実行する必要があります。ただし、通常の場合、インデックスの最上位レベルは、バッファ内に格納されています。

十分な数のバッファがある場合は、下のレベルもバッファ内に格納されます。ソート・インデックス DEPARTMENTS_INDEX（例 4-32（4-160））については、最大 I/O パスの長さは

1 ～ 2、平均 I/O パスの長さは 1 ～ 1.65 です。この値から、データ行の読取りでは、最大 2 ページまでにアクセスすることがわかります。DEPARTMENTS_INDEX を使うと、ソート・インデックス構造とデータ行は 2 つの隣り合ったページ上に格納されているため、最上位レベル・ノードのレコード、最下位レベル・ノードのレコード、データ行は、1 回の I/O 操作で取り出されます。

ソート・インデックスのさらに低いレベルを含めて、上記のレコードがすべてバッファ内に存在していることを確認するには、RMU Analyze Placement コマンドに Option=Full 修飾子を指定して実行し、頻度のヒストグラムで MIN I/O パスの長さを調べてください。最小 I/O パスの長さの頻度分布では、データ行へのアクセス性が、データ・アクセスに必要な I/O 操作の数で示されています。最小 I/O パスの長さが 1 よりも大きい場合、最上位レベル、最下位レベル、データ行を 1 回の I/O 操作で読み取ることはできません。

SQL の CREATE および ALTER STORAGE MAP 文の PLACEMENT VIA INDEX 句を使う場合、データと同じ記憶領域の同じページ上にあるハッシュ・インデックスでは、データ行の読取りに、最低 1 回の I/O 操作が必要です。

オーバーフローが発生すると、データ行と同じページ上にあるハッシュ・バケットに空きがあればデータ行の書込みは可能ですが、空き領域が不足してエントリを作成できない場合、ハッシュ・バケットは隣のページにオーバーフローします。データ行を追加すると、データ・ページ全体をハッシュしようとして隣のページに書込みが実行され、両方のレコード型を書き込む領域が存在する場合は、オーバーフロー・ハッシュ・バケットにエントリが追加されます。その結果、ハッシュ・バケットとデータは異なるページに格納されるので、最大 I/O パスの長さは、ハッシュ・バケットとデータ行が書き込まれたページ数だけ長くなります。

重複が許可されている場合は、重複ノード・レコードが作成され、ハッシュ構造の一部としてデータ・ページに格納されます。重複レコードが多数あり、ページ・サイズが小さすぎる場合、データ行は使用可能領域がある隣接ページに格納され、重複ノード・レコードも追加されるので、最終的には、オーバーフロー・ハッシュ・バケットのエントリも増大します。時間と伴にデータ・ページはいっぱいになり、オーバーフローが頻繁に発生すると、最小 I/O パスの長さは、ハッシュ・インデックス構造とデータ行全体が同じバッファ内にないことを示すので、最大 I/O パスの長さは増大し、読取りに必要な I/O が増加して、パフォーマンスは低下します。

JOB_HISTORY_HASH ハッシュ・インデックス（例 4-34（4-161））は、重複レコードを含み、最大 I/O パスの長さは 1 ～ 3 ですが、平均の長さは 1.00 ～ 1.08 になっています。これは、最大 I/O パスの長さが 1 ページを超えるデータ行は、12 のうちわずか 1 ページであることを示します。この例では、ハッシュ構造とデータ行は 1 回の I/O 操作ですべて取り出すことができます。

重複レコードがすべてバッファ内に存在していることを確認するには、RMU Analyze Placement コマンドに Option=Full 修飾子を指定して実行し、頻度のヒストグラムで MIN I/O パスの長さを調べてください。最小 I/O パスの長さの頻度分布では、データ行へのアクセス性が、データ・アクセスに必要な I/O 操作の数で示されています。最小 I/O パスの長さが 1 よりも大きい場合、すべてのハッシュ構造（システム・レコード、ハッシュ・バケッ

ト、オーバーフロー・ハッシュ・バケット、重複ノード・レコード）とデータ行は、すべてを 1 回の I/O 操作で読み取ることはできません。

4.3.1.2 RMU Analyze Placement Option=Full コマンドの使用方法

RMU Analyze Placement コマンドで Option=Full 修飾子を指定すると、Oracle RMU は、次のような情報を表示します。

- 概要情報
  概要情報は、インデックスと Option=Normal 修飾子を指定した場合と同じ内容を表示します。
- DBkey パスの長さと頻度ヒストグラム
  DBkey パスの長さの頻度ヒストグラムは、データ行へのアクセスに必要な DBkey の数の頻度分布を示します。
- MAX I/O パスの長さと頻度ヒストグラム
  MAX I/O パスの長さの頻度ヒストグラムは、データ行へのアクセスに必要な最大ページ数の頻度分布を示します。
- MIN I/O パスの長さと頻度ヒストグラム
  MIN I/O パスの長さと頻度ヒストグラムは、インデックスとデータ行の両方が同時にバッファ内に存在する比率を、各バッファ・サイズに応じた頻度分布を示します。

EMPLOYEES_HASH ハッシュ・インデックスについては、インデックスは 1 レベルのみです。例 4-35（4-163）は、EMPLOYEES_HASH ハッシュ・インデックスの情報を示します。

例 4-35 ハッシュ・インデックスでの RMU Analyze Placement Option=Full コマンド

```
$ RMU/ANALYZE/PLACEMENT mf_personnel EMPLOYEES_HASH /OPTION=FULL
-------------------------------------------------------------------------------
 Indices for database  - $DUA0:[ORION]MF_PERSONNEL.RDB;
-------------------------------------------------------------------------------
 Hashed index EMPLOYEES_HASH for relation EMPLOYEES duplicates not allowed
 Levels: 1, Nodes: 69, Keys: 100, Records: 100
 Maximum path length --  DBkeys: 3, IO range: 1 to 1
 Average path length --  DBkeys: 3.00, IO range: 1.00 to 1.00
                       DBkey path length vs. frequency
    6 | (0)
    5 | (0)
    4 | (0)
    3 |================================================== (100)
    2 | (0)
    1 | (0)
                     MAX IO path length vs. frequency
    6 | (0)
    5 | (0)
    4 | (0)
```

```
    3 | (0)
    2 | (0)
    1 |================================================== (100)
                    MIN IO path length vs. frequency
    6 | (0)
    5 | (0)
    4 | (0)
    3 | (0)
    2 | (0)
    1 |================================================== (100)
-------------------------------------------------------------------------------
```

DEPARTMENTS_INDEX ソート・インデックスについては、インデックスは 1 レベルのみです。例 4-36（4-164）は、DEPARTMENTS_INDEX ソート・インデックスの情報を示します。

例 4-36 ソート・インデックスでの RMU Analyze Placement Option=Full コマンド

```
$ RMU/ANALYZE/PLACEMENT mf_personnel DEPARTMENTS_INDEX /OPTION=FULL
-------------------------------------------------------------------------------
 Indices for database  - $DUA0:[ORION]MF_PERSONNEL.RDB;
-------------------------------------------------------------------------------
 Index DEPARTMENTS_INDEX for relation DEPARTMENTS duplicates not allowed
 Levels: 1, Nodes: 1, Keys: 26, Records: 26
 Maximum path length --  DBkeys: 2, IO range: 1 to 2
 Average path length --  DBkeys: 2.00, IO range: 1.00 to 1.65
                     DBkey path length vs. frequency
    6 | (0)
    5 | (0)
    4 | (0)
    3 | (0)
    2 |================================================== (26)
    1 | (0)
                    MAX IO path length vs. frequency
    6 | (0)
    5 | (0)
    4 | (0)
    3 | (0)
    2 |================================================== (17)
    1 |========================== (9)
                    MIN IO path length vs. frequency
    6 | (0)
    5 | (0)
    4 | (0)
    3 | (0)
    2 | (0)
    1 |================================================== (26)
-------------------------------------------------------------------------------
```

例 4-37（4-165）は、JOB_HISTORY_HASH ハッシュ・インデックスの情報を示します。

例 4-37 ハッシュ・インデックスでの RMU Analyze Placement Option=Full コマンド（重複可）

```
$ RMU/ANALYZE/PLACEMENT mf_personnel JOB_HISTORY_HASH /OPTION=FULL
-------------------------------------------------------------------------------
 Indices for database  - $DUA0:[ORION]MF_PERSONNEL.RDB;
-------------------------------------------------------------------------------
 Hashed index JOB_HISTORY_HASH for relation JOB_HISTORY duplicates allowed
 Levels: 1, Nodes: 69, Keys: 100, Records: 274
   Dup nodes: 80, Dup keys: 80, Dup records: 254
 Maximum path length --  DBkeys: 4, IO range: 1 to 3
 Average path length --  DBkeys: 3.93, IO range: 1.00 to 1.08
                        DBkey path length vs. frequency
     6 | (0)
     5 | (0)
     4 |================================================== (254)
     3 |==== (20)
     2 | (0)
     1 | (0)
                       MAX IO path length vs. frequency
     6 | (0)
     5 | (0)
     4 | (0)
     3 | (1)
     2 |==== (21)
     1 |================================================== (252)
                       MIN IO path length vs. frequency
     6 | (0)
     5 | (0)
     4 | (0)
     3 | (0)
     2 | (0)
     1 |================================================== (274)
-------------------------------------------------------------------------------
```

3 つのインデックス EMPLOYEES_HASH、DEPARTMENTS_INDEX および JOB_HISTORY_HASH は、ヒストグラムを使うと、次のようにさらに詳細に分析できます。

- EMPLOYEES_HASH ハッシュ・インデックス（例 4-35（4-163））
  100 あるデータ行すべてにおいて、DBkey パスの長さは 3 です（システム・レコード、ハッシュ・バケット、データ行）。100 のインデックス・レコードについて、最大 I/O パスの長さ（アクセスしたページ数）は 1 です。これは、EMPLOYEES_HASH ハッシュ・バケットが、EMPLOYEES データ行と同じページを共有していることを意味します。100 のインデックス・レコードにおいて、最小 I/O パスの長さは 1 です。これは、バッファ・サイズを考えると、どの EMPLOYEES_HASH ハッシュ・バケットについても、EMPLOYEES データ行を 1 回の I/O 操作で取り出せるということを意味します。

- DEPARTMENTS_INDEX ソート・インデックス（例 4-36（4-164））
  26 あるデータ行すべてにおいて、DBkey パスの長さは 2 です（リーフ・ノードとデータ行）。26 すべてのインデックス・レコードについては、最大 I/O パスの長さ（アクセスしたページ数）は、17 のインデックス・レコードで 2、9 のインデックス・レコードでは 1 です。つまり、9 つの DEPARTMENTS_INDEX リーフ・ノード・レコードは、DEPARTMENTS データ行と同じページを共有し、17 の DEPARTMENTS データ行へのアクセスは、2 ページを横断していることを意味します。26 のインデックス・レコードにおいて、最小 I/O パスの長さは 1 です。これは、バッファ・サイズを考えると、どの DEPARTMENTS_INDEX リーフ・ノード・レコードについても、DEPARTMENTS データ行を 1 回の I/O 操作で取り出せるということを意味します。

- JOB_HISTORY_HASH ハッシュ・インデックス（例 4-37（4-165））
  274 のデータ行での DBkey パスの長さは、20 のインデックス・レコード（システム・レコード、ハッシュ・バケット、データ行）では 3、254 のレコード（システム・レコード、ハッシュ・バケット、重複ノード・レコード、データ行）では 4 です。274 のインデックス・レコードでの最大 I/O パスの長さ（アクセスしたページ数）は、252 のインデックス・レコードで 1、21 のインデックス・レコードで 2、1 つのインデックス・レコードで 3 です。つまり、ほとんどの JOB_HISTORY_HASH ハッシュ・バケット（252）が JOB_HISTORY データ行と同じページを共有、21 の行へのアクセスには 2 ページを横断、1 つのデータ行のアクセスには 3 ページを横断していることがわかります。274 のインデックス・レコードにおいて、最小 I/O パスの長さは 1 です。これは、バッファ・サイズを考えると、どの JOB_HISTORY ハッシュ・バケットについても、JOB_HISTORY データ行を 1 回の I/O 操作で取り出せるということを意味します。JOB_HISTORY_HASH ハッシュ・インデックスでは重複が許可されているので、重複のない 20 のデータ行へのアクセスは 3 DBkey、重複ノード・レコードを含めた 254 のデータ行へのアクセスは 4 DBkey で実行できます。
  ページには、ハッシュ・インデックス構造と、それに対応する親子データ行が格納されているため、最大 I/O パスの長さから、JOB_HISTORY データ行と同じページ上にないハッシュ・バケットの比率はわずかであることがわかります（8%、つまり 22 のインデックス・レコード）。DBkey パスの長さの頻度ヒストグラムと MAX I/O パスの長さの頻度ヒストグラムを時間の経過を追って調べると、特に DBkey が 5 以上で DBkey パスの長さの頻度が増大し、DBkey が 2 以上で最大 I/O パスの長さが増大していく様子がわかります。また、さらに重要なのは、MIN I/O パスの長さの頻度ヒストグラムを調べると、ハッシュ構造とデータ行を 1 回の I/O 操作で読み取ることができるかどうかがわかります。最小 I/O パスの長さが 1 よりも大きくなると、インデックスとデータ行の両方を取り出すために必要な I/O の数が増えるので、パフォーマンスはある程度低下します。

4.3.1.3 RMU Analyze Placement Option=Debug コマンドの使用方法

RMU Analyze Placement コマンドで Option=Debug 修飾子を指定すると、Oracle RMU は、次のようなインデックス・ノードとインデックス・レコードに関する詳細情報を表示します。

- データ・ページ上のデータ行の分布
- データ・ページ上にあるデータ行に達するまでに必要なキーの数
- データ行にアクセスするまでの最大 I/O パスの長さ（横断するページ数）
- バッファ・サイズに関して、行にアクセスするための最小 I/O パスの長さ
- 各キーの長さと、データ行に指定されたキー

出力のヘッダーは、一般情報と詳細情報の 2 つに区分されています。ヘッダーの後には、詳細なインデックス・レコード情報が表示されます。最後には、Output=Full 修飾子を指定した RMU Analyze Placement コマンドと同じ内容と、3 種類のヒストグラムが表示されます。インデックスが複数の論理領域に格納されている場合、すべての論理領域が表示されます。

例 4-38（4-167）は、Option=Debug 修飾子で RMU Analyze Placement コマンドを使用し、EMPLOYEES_HASH ハッシュ・インデックスを指定した場合の出力内容を示しています。

例 4-38 ハッシュ・インデックスでの RMU Analyze Placement Option=Debug コマンド

```
$ RMU/ANALYZE/PLACEMENT mf_personnel EMPLOYEES_HASH /OPTION=DEBUG
0------------------------------------------------------------------------------
0
0 Indices for database  - $DUA0:[ORION]MF_PERSONNEL.RDB;
0
0   ID   Hash Dup Name                              Relation
0------------------------------------------------------------------------------
4      62   T   F  EMPLOYEES_HASH                  EMPLOYEES
4      60   T   F  EMPLOYEES_HASH                  EMPLOYEES
4      58   T   F  EMPLOYEES_HASH                  EMPLOYEES
0
0------------------------------------------------------------------------------
0
0 Selected areas -
0 Area =   1 - RDB$SYSTEM
0 Area =   2 - EMPIDS_LOW
0 Area =   3 - EMPIDS_MID
0 Area =   4 - EMPIDS_OVER
0 Area =   5 - DEPARTMENTS
0 Area =   6 - SALARY_HISTORY
0 Area =   7 - JOBS
0 Area =   8 - EMP_INFO
0 Area =   9 - RESUME_LISTS
0 Area =  10 - RESUMES
0
0------------------------------------------------------------------------------
0
0   ID        DB KEY      DB_KEYS  MAX_IO  MIN_IO (LEN)"KEY"
0------------------------------------------------------------------------------
```

```
0
7  58 0063:0000000002:001       3      1       1 (  6)"003030313635"
7  58 0063:0000000002:003       3      1       1 (  6)"003030313930"
7  58 0063:0000000005:001       3      1       1 (  6)"003030313837"
7  58 0063:0000000007:001       3      1       1 (  6)"003030313639"
7  58 0063:0000000007:003       3      1       1 (  6)"003030313736"
7  58 0063:0000000007:004       3      1       1 (  6)"003030313938"
.
.
.
7  60 0064:0000000003:001       3      1       1 (  6)"003030323133"
7  60 0064:0000000004:001       3      1       1 (  6)"003030323139"
7  60 0064:0000000005:001       3      1       1 (  6)"003030323235"
7  60 0064:0000000005:003       3      1       1 (  6)"003030323430"
7  60 0064:0000000006:001       3      1       1 (  6)"003030323637"
7  60 0064:0000000009:001       3      1       1 (  6)"003030323837"
.
.
.
7  62 0065:0000000008:001       3      1       1 (  6)"003030343135"
7  62 0065:0000000018:001       3      1       1 (  6)"003030343335"
7  62 0065:0000000021:001       3      1       1 (  6)"003030343035"
7  62 0065:0000000026:001       3      1       1 (  6)"003030343138"
7  62 0065:0000000032:001       3      1       1 (  6)"003030343731"
7  62 0065:0000000046:001       3      1       1 (  6)"003030343136"
-------------------------------------------------------------------------------
 Indices for database  - $DUA0:[ORION]MF_PERSONNEL.RDB;
-------------------------------------------------------------------------------
0 Hashed index EMPLOYEES_HASH for relation EMPLOYEES duplicates not allowed
0 Levels: 1, Nodes: 69, Keys: 100, Records: 100
0 Maximum path length --  DBkeys: 3, IO range: 1 to 1
0 Average path length --  DBkeys: 3.00, IO range: 1.00 to 1.00
.
.
.
```

次は、Option=Debug 修飾子を指定した RMU Analyze Placement コマンドのフィールド一覧です。フィールドの説明の後に続くカッコで囲んだ内容は、例 4-38（4-167）で示した EMPLOYEES_HASH インデックス、論理領域 58 に対応しています。

- インデックスの一般情報
  - ID
    インデックスの論理領域 ID（58）
  - Hash
    コード化されたフィールドハッシュ・インデックスでは、T = TRUE、B ツリー・インデックスでは F = FALSE、（T）

- Dup
  コード化されたフィールド重複可なら T = TRUE、重複不可なら F = FALSE、(F)
- Name
  インデックスの名前（EMPLOYEES_HASH）
- Relation
  インデックスを使用するテーブルの名前（EMPLOYEES）

- 選択した領域の凡例
  mf_personnel データベースを構成する記憶領域を示す判例
- インデックスの詳細情報
  - ID
    インデックスの論理領域 ID（58）
  - DB KEY
    レコードの dbkey で、論理領域 ID（0063）、ページ番号（0000000002）、レコードが格納されているページの行（001）の 3 つで構成
  - DB_KEYS
    データ行にアクセスするのに必要な dbkey の数（3）。システム・レコード、ハッシュ・バケット、データ行 dbkey
  - MAX_IO
    最大 I/O パスの長さ、つまり、データ行にアクセスするために横断するページの合計数（1）
  - MIN_IO
    最小 I/O パスの長さ 1 は、インデックスとデータ行の両方がバッファ内に存在することを示す（1）
  - (LEN)
    キーの長さをバイト単位で表示（6）
  - "KEY"
    実際のキーを 16 進数文字列で表示（003030313635）

Option=Debug 修飾子で RMU Analyze Placement コマンドを使用し、DEPARTMENTS_INDEX ソート・インデックスを指定すると、Oracle RMU は、例 4-39（4-169）のような内容を出力します。

例 4-39 ソート・インデックスでの RMU Analyze Placement Option=Debug コマンド

```
$ RMU/ANALYZE/PLACEMENT mf_personnel DEPARTMENTS_INDEX /OPTION=DEBUG
0------------------------------------------------------------------------------
0
0 Indices for database  - $DUA0:[ORION]MF_PERSONNEL.RDB;
```

```
0
0   ID   Hash Dup Name                            Relation
0-------------------------------------------------------------------------------
4      72   F   F  DEPARTMENTS_INDEX              DEPARTMENTS
0
0-------------------------------------------------------------------------------
0
0 Selected areas -
0 Area =   1 - RDB$SYSTEM
0 Area =   2 - EMPIDS_LOW
0 Area =   3 - EMPIDS_MID
0 Area =   4 - EMPIDS_OVER
0 Area =   5 - DEPARTMENTS
0 Area =   6 - SALARY_HISTORY
0 Area =   7 - JOBS
0 Area =   8 - EMP_INFO
0 Area =   9 - RESUME_LISTS
0 Area =  10 - RESUMES
0
0-------------------------------------------------------------------------------
0
0   ID        DB KEY      DB_KEYS  MAX_IO  MIN_IO (LEN)"KEY"
0-------------------------------------------------------------------------------
0
7  72 0073:0000000002:001       2      1       1 (  5)"0041444D4E"
7  72 0073:0000000002:003       2      1       1 (  5)"00454C454C"
7  72 0073:0000000002:004       2      1       1 (  5)"00454C4753"
7  72 0073:0000000002:005       2      1       1 (  5)"00454C4D43"
7  72 0073:0000000002:006       2      1       1 (  5)"00454E4720"
7  72 0073:0000000002:007       2      1       1 (  5)"004D424D46"
7  72 0073:0000000002:008       2      1       1 (  5)"004D424D4E"
7  72 0073:0000000002:009       2      1       1 (  5)"004D424D53"
7  72 0073:0000000003:001       2      2       1 (  5)"004D43424D"
7  72 0073:0000000003:002       2      2       1 (  5)"004D434253"
7  72 0073:0000000003:003       2      2       1 (  5)"004D475654"
7  72 0073:0000000003:004       2      2       1 (  5)"004D4B5447"
7  72 0073:0000000003:005       2      2       1 (  5)"004D4E4647"
7  72 0073:0000000003:006       2      2       1 (  5)"004D534349"
7  72 0073:0000000003:007       2      2       1 (  5)"004D534D47"
7  72 0073:0000000003:008       2      2       1 (  5)"004D54454C"
7  72 0073:0000000003:009       2      2       1 (  5)"005045524C"
7  72 0073:0000000003:010       2      2       1 (  5)"0050455253"
7  72 0073:0000000003:011       2      2       1 (  5)"005048524E"
7  72 0073:0000000003:012       2      2       1 (  5)"0050524D47"
7  72 0073:0000000003:013       2      2       1 (  5)"0053414C45"
7  72 0073:0000000003:014       2      2       1 (  5)"0053455552"
7  72 0073:0000000003:015       2      2       1 (  5)"0053554E45"
```

```
7  72 0073:0000000003:016       2      2       1 (  5)"0053555341"
7  72 0073:0000000003:017       2      2       1 (  5)"005355534F"
7  72 0073:0000000002:010       2      1       1 (  5)"0053555745"
0------------------------------------------------------------------------------
0
0 Indices for database  - $DUA0:[ORION]MF_PERSONNEL.RDB;
0
0------------------------------------------------------------------------------
0 Index DEPARTMENTS_INDEX for relation DEPARTMENTS duplicates not allowed
0 Levels: 1, Nodes: 1, Keys: 26, Records: 26
0 Maximum path length --  DBkeys: 2, IO range: 1 to 2
0 Average path length --  DBkeys: 2.00, IO range: 1.00 to 1.65
.
.
.
```

例 4-39（4-169）では、DEPARTMENTS_INDEX ソート・インデックスは、DEPARTMENTS テーブルに設定されています。インデックス全体（26 レコード）は、論理領域 72 のページ 2 と 3 に格納されています。最初の dbkey である 0073:0000000002:001 は、ポイント先のデータ行が論理領域 73、ページ 2、ライン・エントリ 1 にあることを示しています。データ行にアクセスするのに必要な dbkey の合計は 2 です（リーフ・ノードとデータ行）。最大 I/O パスの長さ（アクセスしたページ数）は 1 です。最小 I/O パスの長さ（バッファを考慮）は 1 です。キーの長さは 5 バイトで、キーの値を 16 進数文字列で表示すると、"0041444D4E" です。

ソート・インデックスでは、インデックス・ノードが小さく重複がない場合、テーブルでインデックスを維持するために、システム・リソースはほとんど使用しません。

一般的には、インデックス・ノードの長さと、重複可能であれば重複の数に注意してください。インデックスが長い場合は、テーブルでインデックスを維持するために、大量のシステム・リソースが消費されている可能性があります。さらに、重複可能で、インデックスに重複した値が多数存在する場合、インデックスの効率も低下します。

Option=Debug 修飾子を使用することにより、問題が発生している箇所を特定できます。最初は、Option=Full 修飾子を使用するときに問題が現れます。dbkey、最大 I/O パスの長さ、最小 I/O パスの長さなどの値を定期的にチェックし、変化や傾向に注意してください。

RMU Analyze Placement コマンドを Option=Debug 修飾子を指定し、JOB_HISTORY_HASH ハッシュ・インデックスを指定すると、Oracle RMU は、例 4-40（4-172）のような内容を出力します。

例 4-40 ハッシュ・インデックスでの RMU Analyze Placement Option=Debug コマンド（重複可）

```
$ RMU/ANALYZE/PLACEMENT mf_personnel JOB_HISTORY_HASH /OPTION=DEBUG
0------------------------------------------------------------------------------
0
0 Indices for database  - $DUA0:[ORION]MF_PERSONNEL.RDB;
0
0   ID   Hash Dup Name                            Relation
0------------------------------------------------------------------------------
4      68   T   T  JOB_HISTORY_HASH               JOB_HISTORY
4      67   T   T  JOB_HISTORY_HASH               JOB_HISTORY
4      66   T   T  JOB_HISTORY_HASH               JOB_HISTORY
0
0------------------------------------------------------------------------------
0
0 Selected areas -
0 Area =   1 - RDB$SYSTEM
0 Area =   2 - EMPIDS_LOW
0 Area =   3 - EMPIDS_MID
0 Area =   4 - EMPIDS_OVER
0 Area =   5 - DEPARTMENTS
0 Area =   6 - SALARY_HISTORY
0 Area =   7 - JOBS
0 Area =   8 - EMP_INFO
0 Area =   9 - RESUME_LISTS
0 Area =  10 - RESUMES
0
0------------------------------------------------------------------------------
0
0   ID        DB KEY      DB_KEYS  MAX_IO  MIN_IO (LEN)"KEY"
0------------------------------------------------------------------------------
7  66 0069:0000000002:009       4      1       1 (  6)"003030313635"
7  66 0069:0000000002:008       4      1       1 (  6)"003030313635"
7  66 0069:0000000002:006       4      1       1 (  6)"003030313635"
7  66 0069:0000000002:004       4      1       1 (  6)"003030313635"
7  66 0069:0000000002:014       4      1       1 (  6)"003030313930"
7  66 0069:0000000002:013       4      1       1 (  6)"003030313930"
7  66 0069:0000000002:011       4      1       1 (  6)"003030313930"
7  66 0069:0000000002:010       4      1       1 (  6)"003030313930"
7  66 0069:0000000005:006       3      1       1 (  6)"003030313837"
7  66 0069:0000000007:010       4      1       1 (  6)"003030313639"
7  66 0069:0000000007:009       4      1       1 (  6)"003030313639"
7  66 0069:0000000007:007       4      1       1 (  6)"003030313639"
7  66 0069:0000000007:005       4      1       1 (  6)"003030313639"
7  66 0069:0000000005:005       4      2       1 (  6)"003030313736"
.
```

```
.
.
7  67 0070:0000000003:003       3      1       1 (  6)"003030323133"
7  67 0070:0000000004:007       4      1       1 (  6)"003030323139"
7  67 0070:0000000004:005       4      1       1 (  6)"003030323139"
7  67 0070:0000000004:003       4      1       1 (  6)"003030323139"
.
.
.
7  68 0071:0000000008:008       4      1       1 (  6)"003030343135"
7  68 0071:0000000008:007       4      1       1 (  6)"003030343135"
7  68 0071:0000000008:005       4      1       1 (  6)"003030343135"
7  68 0071:0000000008:003       4      1       1 (  6)"003030343135"
7  68 0071:0000000018:007       4      1       1 (  6)"003030343335"
-------------------------------------------------------------------------------
 Indices for database  - $DUA0:[ORION]MF_PERSONNEL.RDB;
-------------------------------------------------------------------------------
0 Hashed index JOB_HISTORY_HASH for relation JOB_HISTORY duplicates allowed
0 Levels: 1, Nodes: 69, Keys: 100, Records: 274
0   Dup nodes: 80, Dup keys: 80, Dup records: 254
0 Maximum path length --  DBkeys: 4, IO range: 1 to 3
0 Average path length --  DBkeys: 3.93, IO range: 1.00 to 1.08
.
.
.
```

例 4-40（4-172）の JOB_HISTORY_HASH ハッシュ・インデックスでは、テーブル JOB_HISTORY のインデックスは、3 つの論理領域（66、67、68）に分割されています。最初の dbkey である 0069:0000000002:009 は、ポイント先のデータ行は、論理領域 69、ページ 2、ライン・エントリ 9 にあることを示しています。このデータ行にアクセスするのに必要な dbkey の合計は 4 です（システム・レコード、ハッシュ・バケット、重複ノード・レコード、データ行）。最大 I/O パスの長さ（アクセスしたページ数）は 1 です。最小 I/O パスの長さ（バッファを考慮）は 1 です。キーの長さは 6 バイトで、キーの値を 16 進数文字列で表示すると、"003030313635" です。

JOB_HISTORY_HASH ハッシュ・インデックスでは重複が許可されているので、データ行にアクセスするには重複ノード・レコードにアクセスする必要があり、dbkey の数は 4 に増えます。この場合、最大 I/O パスの長さは 2 に増えます。このことから、データ行は、ハッシュ・バケットまたは重複ノード・レコードとは異なるページ上にあることがわかります。

ページの空き領域がなくなってきたら、最小 I/O パスの長さをモニターしてください。値が 1 を超えると、インデックス・レコードとデータ行は両方ともバッファ内に存在しなくなるので、データ行へのアクセスには、さらに多くの I/O が必要になります。これは、パフォーマンス低下にみられる兆候です。

ハッシュ・インデックスでは、ハッシュ・バケットのオーバーフローが発生したときは、特に新しい親子の重複レコードが多数追加された場合には、詳細情報を定期的にモニターし、ハッシュ・バケットのオーバーフローの特徴や程度を把握してください。

ハッシュ・バケット・オーバーフローの最初の兆候は、RMU Analyze Indexes コマンドまたは RMU Analyze Placement コマンドを実行した結果、最大インデックス・レベルが 1 から 2 以上に増加したときです。また、どのデータ行でも最小 I/O パスの長さは 1 のはずですが、これが 1 を超えた場合には、注意が必要です。最小 I/O パスの長さが 2 以上であることは、ハッシュ・インデックス・レコードとデータ行は物理的に近い位置には格納されていないので、1 回の I/O 操作ではアクセスできず、2 回以上の I/O 操作が必要であることを示します。これは、データベース・アプリケーションのパフォーマンス低下につながります。

PLACEMENT VIA INDEX オプションを使用する場合は、ページ・サイズと、最初にページに格納する子の重複レコードの数に注意してください。親と子両方のデータ行と、定義されたハッシュ構造を同じページ上に格納するには、最初に十分な領域をページ・サイズとして選択し、将来的なレコードの追加にも対応できる領域を確保してください。

問題の特定を目的としたユーザー定義のレポートを作成するには、Option=Debug 修飾子を指定した RMU Analyze Placement コマンドからの出力をプログラムの入力として使用します。たとえば、所定の値を上回っている dbkey や、最小 I/O パスの長さが 1 を超えるものがないか、コマンド出力をスキャンし、該当するレコード全体を印刷します。Option=Full 修飾子を指定した RMU Analyze Placement コマンドでは、dbkey パスの長さ、最大 I/O パスの長さ、最小 I/O パスの長さを表す頻度の分布情報を、3 種類のヒストグラムで表します。アクセスに関する問題の兆候は、このヒストグラムで十分確認できます。

4.3.2 PLACEMENT VIA INDEX オプション

PLACEMENT VIA INDEX オプションでは、SQL の CREATE STORAGE MAP 文で指定したテーブルに行を格納するために使用するインデックスの名前を指定できます。行は、STORE 句の指定に従って、記憶領域に格納されます。配置インデックスを使用すると、最初に行を指定の順番で格納するので、行へのアクセスが効率的になります。新しい行の格納先となるデータ・ページに十分な領域が存在する限り、このようなパフォーマンス向上効果は持続します。必要なページおよびファイルの割当てサイズを慎重に計算し、将来的にもデータ・ページに十分な領域を確保してください。たとえば、複合記憶領域を使用し、ハッシュされた配置インデックスと適正サイズを設定したページおよびファイル割当てに、インデックス構造とデータ行を格納した場合、完全一致クエリーにおいて、1 回の I/O 操作で行を取り出すことができます。

データ・ページが行でいっぱいになり、隣のページにオーバーフローし始めると、インデックスと PLACEMENT VIA INDEX オプションによるパフォーマンス向上効果は低下していきます。パフォーマンスを復帰させるには、SQL の ALTER DATABASE 文を使用して、同等の記憶領域を新しく定義してください。SQL の ALTER STORAGE MAP 文を変更し、各ストレージ・マップが新しい記憶領域をポイントするように設定すると、効率の低い記憶領

域から新しい記憶領域にデータが再ロードされます。詳細は、『Oracle Rdb7 Guide to Database Maintenance』を参照してください。

4.3.3 テーブルのデータ圧縮オプション

データ圧縮を行うと、Oracle Rdb は、より多くのデータを少ないディスク・ブロックに格納できます。領域を節約でき、データの取出しにかかる時間も短縮できます。新しいテーブルの作成では、SQL の CREATE STORAGE MAP 文を使用して、ENABLE COMPRESSION オプションを指定すれば、データ圧縮を設定できます。SQL の ALTER STORAGE MAP 文を使用して、テーブル名と DISABLE COMPRESSION オプションを指定すれば、STORE 文で指定した記憶領域にあるテーブル行のデータ圧縮が自動的に変更されます。また、別の方法として、SQL の EXPORT 文と IMPORT 文を使用し、データ圧縮の変更や新しい記憶領域の定義、新しい圧縮パラメータの指定を行い、SQL の ALTER STORAGE MAP 文で、新しい記憶領域への行の再マッピングを行います。特に、データベースを大きく変更する計画があり、一度に導入したい場合には、2 番目の方法を採用してください。

一般的には、データ圧縮を無効にするのは、次の条件に当てはまる場合のみです。

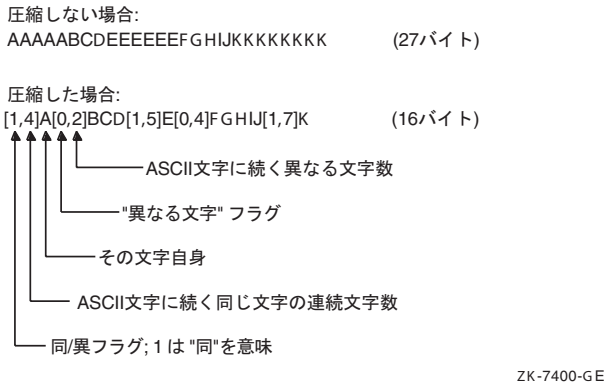
- 多数の行が断片化している
- 行サイズが一定しているので、圧縮によるメリットがない

いずれの場合も、RMU Analyze コマンドを実行し、記憶領域での行の断片化や圧縮による効果を調べてください。RMU Analyz コマンドの使用方法や出力の内容は、2.1 項（2-1）を参照してください。

一般的に、データ圧縮が効果的なのは、頻繁には更新されないテーブルです。このようなテーブルには、かなり固定された情報が格納され、特別なクエリーや順次読取りが主にアクセスします。データを圧縮すると、大量のデータを少ないディスク・ブロックに格納できるので、ディスク領域を節約でき、順次読取りでは、1 回のディスク I/O で多くのデータをユーザー・バッファに読み込めます。

頻繁に更新するテーブルでは、データ圧縮は効果を発揮しません。たとえば、ある注文入力アプリケーションでは、NEXT_ORDER_NUMBER というテーブルがあり、このテーブルには行が 1 つあって、すべてのユーザーがこのテーブルを使用してシーケンス番号を取得します。この行は頻繁に更新されるため、圧縮を行うのは現実的ではありません。圧縮した行を挿入または更新するには、Oracle Rdb はデータ圧縮アルゴリズムを使用して入力をフォーマットする必要があるため、消費する CPU 時間が長くなります。変更やクエリーを非常に頻繁に実行するテーブルでは、行サイズが変更（増加）されると、断片化します。Oracle Rdb が 1 つ以上のデータベース・ページに行を断片化した場合、この行の読出しにかかる時間は大幅に増加します。行のサイズが大幅に変更されている場合は、読出しにはさらに長い時間がかかります。データ圧縮の詳細は、『Oracle Rdb7 SQL Reference Manual』を参照してください。図 4-12（4-176）は、データ圧縮を行う場合と行わない場合のデータを示しています。図では、ASCII コードではなく、文字をそのまま使用しています。

図 4-12 データ圧縮の効果



カッコで囲んだ部分は、1 バイトです。カッコは、格納するデータには含まれません。[1,4]A は、次の内容を意味します。

- 1
  後に続く文字列は、3 つ以上連続した文字を含みます。
- 4
  同じ文字が 5 つあります（文字 A）。0 から始まる表記なので、4 は、5 つの文字が存在することを示します。
- A
  ASCII 値

[0,2]BCD は、次の内容を意味します。

- 0
  後に続く文字列には、同じ文字は含まれません。
- 2
  同じ文字が 3 つあります（文字 BCD）。0 から始まる表記です。

データ圧縮が効果的なタイプとそうでないタイプがあるので、各テーブルごとにデータ圧縮を有効または無効にしてください。

行の変更によって行サイズが物理的に長くなると、行は断片化することがあります。データベース・ページの空き領域が不足し、新しいストレージ・セグメントのサイズを格納できない場合、行が断片化します。

注意： SQL VARCHAR データ型の列を含むテーブルでは、データ圧縮を無効にしないでください。SQL VARCHAR データ型の列のデータ圧縮を無効にすると、Oracle Rdb は、列の最大長を格納します。したがって、SQL VARCHAR データ型を 32,000 バイトに定義し、列には空白でない文字を 1000 文字と 31,000 の空白文字を格納した場合、Oracle Rdb は、合計で 32,000 文字を格納します。Oracle Rdb は、SQL VARCHAR データ型の列のカウント部分を無視し、常に最大長として処理します。

4.3.3.1 データ圧縮の設定例

この項では、データ圧縮の設定例を紹介します。

例 4-41（4-177）では、シングル・ファイルのデータベースで、EMPLOYEES_MAP ストレージ・マップを新しく定義し、EMPLOYEES テーブルのデータ圧縮を無効にします。

例 4-41 シングル・ファイル・データベースでのデータ圧縮の設定

```
SQL> CREATE STORAGE MAP EMPLOYEES_MAP
  1>   STORE IN ...
  2>   DISABLE COMPRESSION;
SQL>
SQL> SHOW STORAGE MAPS EMPLOYEES_MAP
     EMPLOYEES
 For Table:             EMPLOYEES
 Compression is:        DISABLED
 Store clause:          STORE IN ...
```

例 4-42（4-177）では、EMPLOYEES テーブルのデータ圧縮を無効化、SALARY_HISTORY テーブルのデータ圧縮を有効化（デフォルト）しています。次に、EMPLOYEES テーブルでデータ圧縮を無効のままにし、SALARY_HISTORY テーブルを有効から無効に変更します。

それには、SQL の ALTER STORAGE MAP 文で、DISABLE COMPRESSION を指定してください。SQL の ALTER STORAGE MAP 文の STORE 句で指定した記憶領域について、元のストレージ・マップの定義文で指定したテーブルに含まれるすべての行のデータ圧縮が自動的に変更されます。例 4-42（4-177）を参照してください。

例 4-42 STORAGE MAP 文によるテーブルのデータ圧縮の設定

```
SQL> ALTER STORAGE MAP SALARY_HISTORY_MAP
  1>   STORE IN SALARY_HISTORY
  2>   DISABLE COMPRESSION;
```

ただし、圧縮していない行を格納できるだけの領域を確保する必要があり、領域が不足すると、行は断片化し、パフォーマンスが低下します。非圧縮行のサイズを基準にしてページ・サイズを定義する場合には、この点が重要です。記憶領域でデータ圧縮を無効にすると、行が断片化することがわかっている場合は、新しい記憶領域を定義し、正しいページ・サイズを指定して、すべての行を移動してください。複合ページ・フォーマットを使用する記憶領

域では、THRESHOLDS ARE オプションを使用して SPAM しきい値に小さい値を設定すれば、ページあたりの行数の初期値を減らすことができます。

行の圧縮を無効にし、ページ・サイズが小さくても断片化が発生しないようにするには、次のような変更を行ってください。

1. SQL の ALTER DATABASE 文の add-storage-area 句で、新しい名前を指定して記憶領域と記憶領域ファイルを新しく作成し、ページ・サイズまたは SPAM 間隔に大きな値を指定します。
2. SQL の ALTER STORAGE MAP 文で SALARY_HISTORY_MAP ストレージ・マップを変更し、STORE 句を変更して新しい記憶領域をポイントします。データ圧縮の有効化 / 無効化を、無効に設定します。上記の変更をすると、SALARY_HISTORY テーブルの行は新しい記憶領域に移動し、ALTER DATABASE、CREATE STORAGE AREA、ALTER STORAGE MAP 文で指定した内容が有効になります。
3. 古い記憶領域を削除します。

例 4-43（4-178）は、詳細な手順を示しています。

例 4-43 新しい記憶領域の定義と大きなページ・サイズの指定、ストレージ・マップの行の非圧縮、古い記憶領域の削除

```
SQL> ALTER DATABASE FILENAME mf_personnel
  1>   ADD STORAGE AREA NEW_SALARY_HISTORY FILENAME new_salary_history.rda
  2>       ALLOCATION IS 25 PAGES
  3>       PAGE SIZE IS 3 BLOCKS
  4>       PAGE FORMAT IS MIXED
  5>       SNAPSHOT FILENAME new_salary_history.snp
  6>       SNAPSHOT ALLOCATION IS 10 PAGES;
SQL>
SQL> ATTACH 'FILENAME mf_personnel';
SQL>
SQL> ALTER STORAGE MAP SALARY_HISTORY_MAP
  1>   STORE IN NEW_SALARY_HISTORY
  2>   DISABLE COMPRESSION;
SQL>
SQL> COMMIT;
SQL> DISCONNECT ALL;
SQL>
SQL> ALTER DATABASE FILENAME mf_personnel
  1>   DROP STORAGE AREA SALARY_HISTORY;
```

データ圧縮の無効化などの変更は、例 4-44（4-179）の SQL プロシージャのように、SQL の EXPORT と IMPORT 文を使って実行できます。バッファ・サイズの変更など、他の設定の変更も計画している場合には便利です。デフォルトは ENABLE COMPRESSION なので、SQL の ALTER STORAGE MAP 文で DISABLE COMPRESSION を指定しない場合は、

EMPLOYEES と SALARY_HISTORY の両方のテーブルについて、DISABLE COMPRESSION を指定する必要があります。

例 4-44 IMPORT 文による Import パラメータ、新しい記憶領域、ストレージ・マップの指定

```
SQL> EXPORT DATABASE FILENAME mf_personnel.rdb INTO intpers_bu.rbr;
SQL> --
SQL> -- Export uses WITH EXTENSIONS by default
SQL> --
SQL> @IMPORT_COMPRESS
--
IMPORT DATABASE FROM intpers_bu.rbr
--
-- Specify import options
--
NO ACL
--
-- Specify root-file parameters
FILENAME newtest.rdb
BUFFER SIZE IS 12 BLOCKS
--
-- Specify storage area parameters
-- Define the new storage areas needed
--
CREATE STORAGE AREA EMPIDS_LOW FILENAME new_empids_low.rda
--
-- Specify needed options for the new storage area
--
  ALLOCATION IS 50 PAGES
  PAGE FORMAT IS MIXED
  SNAPSHOT FILENAME new_empids_low.snp
  SNAPSHOT ALLOCATION IS 10 PAGES
--
CREATE STORAGE AREA NEW_SALARY_HISTORY FILENAME new_salary_history.rda
  ALLOCATION IS 25 PAGES
  PAGE SIZE IS 3 BLOCKS
  PAGE FORMAT IS MIXED
  SNAPSHOT FILENAME new_salary_history.snp
  SNAPSHOT ALLOCATION IS 10 PAGES
   .
   .
   .
-- Specify metadata options by specifying new storage map options with
-- the CREATE STORAGE MAP statement for the EMPLOYEES_MAP and the
-- SALARY_HISTORY_MAP
--
CREATE STORAGE MAP EMPLOYEES_MAP FOR EMPLOYEES
```

```
  STORE USING EMPLOYEE_ID
    IN
    EMPIDS_LOW WITH LIMIT OF ('00200')
    EMPIDS_MID WITH LIMIT OF ('00400')
    OTHERWISE IN EMPIDS_OVER
    PLACEMENT VIA INDEX EMPLOYEES_HASH
  DISABLE COMPRESSION
--
CREATE STORAGE MAP SALARY_HISTORY_MAP FOR SALARY_HISTORY
  STORE IN NEW_SALARY_HISTORY
  DISABLE COMPRESSION
   .
   .
   .
END IMPORT;
```

例 4-45（4-180）は、SQL プロシージャの実行時に IMPORT 文で表示されるメッセージです。

例 4-45 IMPORT 文のメッセージ

```
Exported by Oracle Rdb V7.0-00 Import/Export utility
A component of SQL V7.0-00
Previous name was mf_personnel.rdb
It was logically exported on 28-MAY-1996 13:25
   .
   .
   .
IMPORTing table EMPLOYEES
IMPORTing table SALARY_HISTORY
   .
   .
   .
-- Import complete
```

SQL の SHOW STORAGE MAP 文を使って、テーブルのデータ圧縮を表示します。例 4-46（4-180）を参照してください。SQL の SHOW STORAGE MAP 文は、明示的な DISABLE 句で定義またはインポートされたテーブルで、データ圧縮が無効になっているかどうかを示します。

例 4-46 SHOW 文で圧縮を確認

```
SQL> SHOW STORAGE MAP EMPLOYEES_MAP
     EMPLOYEES_MAP
 For Table:             EMPLOYEES
 Placement Via Index:   EMPLOYEES_HASH
 Compression is:        DISABLED
 Store clause:          STORE USING (EMPLOYEE_ID)
             IN EMPIDS_LOW WITH LIMIT OF ('00200')
             IN EMPIDS_MID WITH LIMIT OF ('00400')
```

```
            OTHERWISE IN EMPIDS_OVER
SQL>
SQL> SHOW STORAGE MAP SALARY_HISTORY_MAP
     SALARY_HISTORY_MAP
 For Table:             SALARY_HISTORY
 Compression is:        DISABLED
 Store clause:          STORE IN SALARY_HISTORY
SQL>
SQL> SHOW STORAGE MAP JOB_HISTORY_MAP
     JOB_HISTORY
 For Table:             JOB_HISTORY
 Placement Via Index:   JOB_HISTORY_HASH
 Compression is:        ENABLED
 Store clause:          STORE USING (EMPLOYEE_ID)
             IN EMPIDS_LOW WITH LIMIT OF ('00200')
             IN EMPIDS_MID WITH LIMIT OF ('00400')
             OTHERWISE IN EMPIDS_OVER
```

4.3.3.2 データ圧縮オプションの概要

データ圧縮オプションは、次のようにまとめられます。

- ENABLE COMPRESSION（デフォルト）
  取出しアプリケーションが頻繁に使用するテーブルには、データ圧縮は有効にしておきます。
- DISABLE COMPRESSION
  頻繁に更新するテーブルにはデータ圧縮を無効にし、特に変更と取出しが頻繁に行われる行では、無効にしてください。
  このオプションは、SQL の CREATE または ALTER STORAGE MAP 文または SQL の IMPORT 文で明示的に指定する必要があります。SQL CREATE STORAGE MAP 文では、新しいストレージ・マップが最初に定義されます。

4.4 Oracle Rdb アプリケーションでの OpenVMS パラメータの調整

OpenVMS VAX OpenVMS Alpha

オペレーティング・システムのパラメータの変更では、1 つの変更が他に与える影響に注意が必要です。システムの使用率があまり高くない時間帯や、ビジネスにほとんど支障を与えない時間帯を選んで、オペレーティング・システムの変更を試してください。

変更する場合は、データベースの停止時間を十分にとり、変更による効果と、他への影響がないか確認してください。オペレーティング・システムやデータベースで使用するプロセス・パラメータを調整するか、コンピュータ・リソースの他のユーザー・ニーズとのバランスを考えてください。

システムおよびプロセス・パラメータの変更では、十分なメモリー容量、ファイルの制限、エンキューの制限、バイト数の制限を設定したら、アプリケーション設計を時間をかけて詳細に調べてください。システムおよびプロセス・パラメータの設定方法の詳細は、『Oracle Rdb7 Installation and Configuration Guide』を参照してください。オペレーティング・システムとユーザー・プロセス・パラメータに十分大きな値が設定されている場合は、オペレーティング・システムのチューニングをそれ以上続けても、あまり効果はありません。他の部分のチューニングによって、大きな効果を上げられる可能性があります。システムのチューニングの詳細は、7.6.1 項（7-7）を参照してください。

OpenVMS Monitor ユーティリティは、システム・パフォーマンスのモニターに便利です。これは、パフォーマンス・モニターをサンプリングするアプリケーションであり、システム稼動時の情報収集と、以前のシステムの稼動状態に関する概要レポートを作成できます。OpenVMS Monitor ユーティリティの詳細は、OpenVMS のドキュメントを参照してください。また、4.4.1 項（4-182）で説明する Performance Monitor の「VM Usage Statistics」画面も使用できます。◆

4.4.1 Performance Monitor の「VM Usage Statistics」画面

「VM Usage Statistics」画面では、VMScluster ノード上のすべてのデータベース・ユーザーによる動的な仮想メモリーの使用率を概要で示します。（仮想メモリーは、各データベース・ユーザーで一意です）。次に、「VM Usage Statistics」画面の例を示します。

```
Node: TRIXIE          Oracle Rdb V7.0-00 Performance Monitor 28-MAY-1996 13:38:09
Rate: 3.00 Seconds             VM Usage Statistics            Elapsed: 03:17:15.56
Page: 1 of 1        RDBVMS_USER1:[LOGAN.V70]MF_PERSONNEL.RDB;1       Mode: Online
------------------------------------------------------------------------------
statistic.........      rate.per.second.............. total....... average......
name..............      max..... cur..... avg....... count....... per.trans....
GET_VM calls                   1        0        1.0        12914        1844.8
FREE_VM calls                  0        0        0.3         3560         508.5
GET_VM kilobytes               0        0        0.4         4462         637.4
FREE_VM kilobytes              4        0        3.5        41393        5913.2
$EXPREG calls                  0        0        0.0            0          0.0
------------------------------------------------------------------------------
Exit Graph Help Menu Options Pause Reset Set_rate Time_plot Unreset Write X_plot
```

画面の各フィールドについては、Performance Monitor のヘルプを参照してください。

4.4.2 OpenVMS システム・パラメータの確認と設定

OpenVMS VAX OpenVMS Alpha

AUTOGEN は、アプリケーションで求められる処理やパフォーマンスにあわせて、パラメータの値とシステム・ファイル・サイズをリセットします。新しく設定した値やファイル・サイズは、次にシステムとブートしたときに有効になります。AUTOGEN の詳細は、OpenVMS のドキュメントのシステム管理に関する部分と、『Oracle Rdb7 Installation and Configuration Guide』を参照してください。

次のようなシステム・パラメータをチェックしてください。

- CHANNELCNT
  Set high:CHANNELCNT = （アプリケーションによって異なる）
  CHANNELCNT は、システムで使用可能な永続的な I/O チャネルの数を示します。各データベース環境で使用している FILLM の最大値よりも大きな値を設定してください。データベース・アプリケーションでは、1 つのファイルをオープンするたびに、1 つのチャネル・カウントが使用されます。たとえば、ユーザーが 1 人のシングル・ファイル・データベースでは、.rdb ファイル、.snp ファイル、ユーザーの .ruj ファイル、.aij ファイル（有効化されている場合）に 1 つずつ使用するので、5 以上を指定します。ソート・ファイルや、SYS$OUTPUT や SYS$INPUT などに割り当てられたファイルなどの一時ファイルも、チャネル・カウントを使用します。
  FILLM の詳細は、4.4.4 項（4-190）を参照してください。

- IRPCOUNT, IRPCOUNTV
  アプリケーションを実行した後に AUTOGEN を実行し、フィードバックに従ってチューニングを行ってください。

  注意：OpenVMS VAX V6.0 と OpenVMS Alpha V1.5 以上では、オペレーティング・システムが上記のパラメータを自動設定します。システム管理者は、パラメータを設定または変更する必要はありません。

  IRPCOUNT には、事前に割り当てた Intermediate Request Packets（IRP）の数を指定します。1 つの IRP に、160 バイトの常駐メモリーが必要です。IRPCOUNT の値が大きすぎると、物理メモリーが無駄になります。拡張パラメータが小さすぎると、頻繁に拡張が発生し、パフォーマンスが低下します。ただし、IRPCOUNT には、IRPCOUNTV を超える値は設定できません。DCL コマンドの SHOW MEMORY/POOL/FULL を使用した結果、使用中の CURRENT IRP の値が INITIAL IRP よりも大きい場合、SYSGEN パラメータに、CURRENT IRP 以上の値を設定してください。
  IRPCOUNTV は、システムが IRPCOUNT の値を自動的に増加するときの上限になります。この値が小さすぎると、非ページング・プール要求は IRPCOUNTV を使用できなくなるので、システム・パフォーマンスは低下します。

- LRPCOUNT, LRPCOUNTV
  アプリケーションを実行した後に AUTOGEN を実行し、フィードバックに従ってチューニングを行ってください。

  注意：OpenVMS VAX V6.0 と OpenVMS Alpha V1.5 以上では、オペレーティング・システムが上記のパラメータを自動設定します。システム管理者は、パラメータを設定または変更する必要はありません。

IRPCOUNT には、事前に割り当てた Large Request Packet（LRP）の数を指定します。1 つの LRP は、576 バイトの常駐メモリーを使用し、これは LRPSIZE パラメータで指定したバイト数に等しい値です。（通常、LRPSIZE は 576 バイトです）。LRPCOUNT の値が大きすぎると、物理メモリーが無駄になります。LRPCOUNT の値が小さすぎる場合は、システムが適切に稼動できるように、必要に応じて自動的に拡張されます。ただし、システムは、LRPCOUNT に LRPCOUNTV を超える値を設定することはできません。DCL コマンドの SHOW MEMORY/POOL/FULL を使用した結果、使用中の CURRENT LRP の値が INITIAL LRP よりも大きい場合、SYSGEN パラメータに、CURRENT LRP 以上の値を設定してください。
LRPCOUNTV は、システムが LRPCOUNT の値を自動的に増加するときの上限になります。この値が小さすぎると、非ページング・プール要求は、このメモリー割当てを使用できなくなるので、システム・パフォーマンスが低下します。

- SRPCOUNT
  Set high:SRPCOUNT = 8192（小規模なシステム）、= 15000（大規模なシステム）

注意：OpenVMS VAX V6.0 と OpenVMS Alpha V1.5 以上では、オペレーティング・システムが上記のパラメータと SRPCOUNTV を自動設定します。システム管理者は、パラメータを設定または変更する必要はありません。

SRPCOUNT には、事前に割り当てた SRP（small request packet）の数を指定します。各 SRP は、96 バイトの常駐メモリーを使用します。Oracle Rdb は、1 つのロックに SRP を 1 つ（96 バイト）使用します。1 つのロックが SRPCOUNT を 1 つ使用するので、SRPCOUNT には、LOCKIDTBL よりも小さい値は設定しないでください。DCL コマンドの SHOW MEMORY/POOL/FULL を使用した結果、使用中の CURRENT SRP の値が INITIAL SRP よりも大きい場合、SYSGEN パラメータに、CURRENT SRP 以上の値を設定してください。
グローバル・バッファを有効にすると、使用するロックの数が増大するので、SRPCOUNT の値も大きくする必要があります。

- SRPCOUNTV
  Set high:SRPCOUNTV = 32768（小規模なシステム）、= 60000（大規模なシステム）
  SRPCOUNTV は、システムが SRPCOUNT の値を自動的に増加するときの上限になります。SYSGEN を使って SRPCOUNT を手動で設定する場合、SRPCOUNTV を上記の値に設定してください（SRPCOUNT の 4 倍）。AUTOGEN を使って SRPCOUNT を設定する場合、OpenVMS は適切な値を設定するので（通常は SRPCOUNT の 4 倍）、SRPCOUNTV には明示的に値を設定しないでください。
  グローバル・バッファを有効にすると、使用するロックの数が増大するので、SRPCOUNTV の値も大きくする必要があります。

- LOCKIDTBL
  Set high:LOCKIDTBL = 8192
  LOCKIDTBL は、システムのロック ID テーブルのエントリ数の初期値を設定します。また、ロックが不足した場合に、ロック ID テーブルを拡張するサイズを指定します。システムは、領域が不足すると、SYSTEM パラメータである LOCKIDTBL の値だけ、ロック ID を自動的に拡張します。システム内の各ロックについて、ロック ID テーブルにエントリが 1 つ作成され、各エントリは 4 バイト使用します。LOCKIDTBL の値を変更したときは、必ず REHASHTBL の値を確認し、必要に応じて変更してください。
  ロックは、Performance Monitor または OpenVMS Monitor ユーティリティ（MONITOR）でモニターできます。
  LOCKIDTBL は、非ページング・プールから割り当てられます。このパラメータに小さい値を設定すると、次のようなエラー・メッセージが表示されます。

  ```
  %SYSTEM-E-NOLOCKID, no lock id available
  ```

  グローバル・バッファを有効にすると、使用するロックの数が増大するので、LOCKIDTBL の値も大きくする必要があります。

- LOCKIDTBL_MAX
  Set high:LOCKIDTBL_MAX = LOCKIDTBL よりも大きな値
  LOCKIDTBL_MAX には、ロック ID テーブルのサイズの上限を指定します。これは、動的なシステム・パラメータであり、2048 以上の値を永続的に設定する必要があります。LOCKIDTBL を 8192 に設定した場合は、LOCKIDTBL_MAX には 8192 を超える値を設定してください。Oracle Rdb をインストールした後、この値を小さくしないでください。このパラメータに小さい値を設定すると、次のようなエラー・メッセージが表示されます。

  ```
  %SYSTEM-E-NOLOCKID, no lock id available
  ```

  グローバル・バッファを有効にすると、使用するロックの数が増大するので、グローバル・バッファの合計数を基準に、LOCKIDTBL_MAX の値も大きくする必要があります。

- NPAGEDYN
  アプリケーションを実行した後に AUTOGEN を実行し、フィードバックに従ってチューニングを行ってください。
  NPAGEDYN は、非ページング動的プールのサイズをバイト単位で定義します。
  NPAGEDYN パラメータは、非ページング・プール・サイズの初期値を設定しますが、プール・サイズは動的に増加します。このパラメータを設定する際は、最初にデフォルト値を使い、DCL コマンドの SHOW MEMORY/POOL/FULL を使って、実際に使用されている領域をモニターしてください。

- NPAGEVIR
  アプリケーションを実行した後に AUTOGEN を実行し、フィードバックに従ってチューニングを行ってください。
  NPAGEVIR は、システムが NPAGEDYN の値を自動的に増加するときの上限になります。この値が小さすぎると、システムがハングすることがあります。このパラメータを

設定する際は、最初にデフォルト値を使い、DCL コマンドの SHOW MEMORY/POOL/FULL を使って、実際に使用されている領域をモニターしてください。

- GBLPAGES
  GBLPAGES には、n に 1396 ページを加算した値以上を設定してください。n は、Oracle Rdb をインストールする前に使用していたシステム・パラメータの値です。
  GBLPAGES には、起動時に割り当てるグローバル・ページ・テーブルのエントリ数を指定します。各グローバル・セクションは、グローバル・ページ・テーブルのエントリを 1 つ使用します。128 エントリごとに、システム・ページ・テーブル・エントリとして、4 バイトが常駐メモリーに加算されます。
  グローバル・バッファを有効にすると、グローバル・セクションのサイズが大きくなるので、GBLPAGES の値も大きくする必要があります。詳細は、4.1.2.12 項（4-48）を参照してください。

  注意：GBLPAGES がデフォルト値を超える場合は、SYSMWCNT の調整が必要です。GBLPAGES に 128 バイト追加するごとに、SYSMWCNT を 1 増やしてください。SYSMWCNT は、システム・ワーキング・セット、ページング動的プール、OpenVMS RMS およびシステム・メッセージ・ファイルの常駐部分の値を設定します。SYSMWCNT は、大きな値を設定するとユーザーのワーキング・セットが不足してしまいますが、小さな値を設定しても、システム・パフォーマンスに重大な影響を与えます。AUTOGEN コマンドを使えば、上記のパラメータは自動的に設定されます。

- GBLSECTIONS
  GBLSECTIONS には、n に 80 セクション加算した値を設定してください。n は、Oracle Rdb をインストールする前に使用していたシステム・パラメータの値です。
  GBLSECTIONS には、起動時にシステム・ヘッダーに割り当てられるグローバル・セクション記述子の数を設定します。1 つのセクションには、32 バイトの常駐メモリーが必要です。
- MAXBUF
  MAXBUF には、1200 バイト以上を設定してください。
  MAXBUF は、バッファ I/O 転送の最大サイズをバイト数で設定します。バッファ I/O 転送用の領域は、永続的に常駐する非ページング動的プールから割り当てられます。
- PROCSECTCNT
  PROCSECTCNT には、32 バイト以上を設定してください。
  PROCSECTCNT は、プロセスに登録できるセクション記述子の数です。セクション記述子により、プロセス・ヘッダーは、32 バイトという固定サイズだけ増加します。この値には、イメージのリンケージ・メモリー割当てマップに従って、実行するセクションに含まれるイメージ・セクションの最大数よりも大きな値を設定してください。

- VIRTUALPAGECNT
  VIRTUALPAGECNT の現在の値を、各アクティブ・データベースについて 2000 ページ加算してください。
  VIRTUALPAGECNT は、1 つのプロセスにマッピング可能な仮想ページの最大数です。128 の仮想ページごとに、システム・ページ・テーブルで 4 バイトの常駐メモリーが必要になります。
  グローバル・バッファを有効にすると、グローバル・セクションのサイズは増大します。プロセスはグローバル・セクションへのマッピングを行うため、グローバル・セクションの増大に伴って、割り当てられている仮想ページの数も増大します。したがって、グローバル・セクションのサイズが大きくなると、VIRTUALPAGECNT の値も大きくする必要があります。詳細は、4.1.2.12 項（4-48）を参照してください。

- WSMAX
  Set high:WSMAX = システム上で必要なワーキング・セットの最大サイズに設定
  WSMAX は、システム上の任意のワーキング・セットの最大ページ数です。WSMAX には、システム上で必要なワーキング・セットの最大サイズを設定します。これは、異機種混在のクラスタ環境で、メモリーは異なっても共通の UAF ファイルを使用する場合に便利です。通常の時分割処理では、デフォルト値で十分ですが、非常に大きな仮想アドレス空間を使用するプログラムでは、ページ・フォルトを軽減するために、かなり大きな値を設定してください。

- DEADLOCK_WAIT
  DEADLOCK_WAIT は、システムがデッドロック検索を開始するまでに、ロック要求が待機しなければならない秒数を定義します。
  VMScluster または非クラスタのいずれかの環境で、競合と更新が大量に発生するマルチユーザー・アプリケーションを実行する場合は、DEADLOCK_WAIT に小さな値を設定してください。VMScluster の各ノードに 1 秒、この合計に 1 秒を加算してください。VMScluster にノードが 10 ある場合、DEADLOCK_WAIT は 11 秒になります。
  DEADLOCK_WAIT を 11 秒に設定すると、ロック要求は、システムがデッドロック検索を開始するまで、11 秒間待機しなければなりません。
  読取り専用タスクの作業負荷は、使用するロック・リソースが少なく、デッドロックは発生しません。したがって、読取り集中型の環境で DEADLOCK_WAIT に小さな値を設定しても大きな影響はなく、必要のないデッドロックのチェックを実行することによって、システム全体のパフォーマンスが低下する可能性があります。このような環境では、デフォルト値の 10 を設定してください。
  ユーザーのデータベース環境で求められるパフォーマンスを発揮するためには、デフォルトとは異なる最適な値を設定することも可能です。MONITOR を使って、システム上のロック統計をモニターしてください。このユーティリティは、デッドロック検索とデッドロック検出の両方のカウントを表示します。オラクル社は、MONITOR の実行結果から、高いパーセンテージでデッドロックが発生していることがわかった場合以外は、DEADLOCK_WAIT をデフォルト値である 10 秒に設定することをお薦めします。デッドロックの発生頻度が高い場合は、10 秒よりも小さい値に設定してください。10 秒に設定した状態でデッドロックが頻繁に発生しない場合は、DEADLOCK_WAIT を 20 秒または

30 秒に設定します。Performance Monitor が表示するロック統計は、指定したデータベースのみが対象であり、システム全体の統計ではないため、Performance Monitor の統計に基づいて DEADLOCK_WAIT をチューニングする際には注意が必要です。DEADLOCK_WAIT パラメータの詳細は、8.4 項（8-43）を参照してください。◆

OpenVMS Alpha

- VCC_FLAGS, VCC_MAXSIZE

OpenVMS Alpha では、Alpha システム上で仮想 I/O キャッシュを使用し、I/O ボトルネックの軽減とパフォーマンスの向上を実現します。仮想 I/O キャッシュを使用できるのは、シングル・ノード・システムのみであり、クラスタ・システムでは仮想 I/O キャッシュは使用しません。

OpenVMS Alpha Version 1.5 にアップグレードした後で SHOW SYSTEM コマンドを使用すると、多数のプロセスがスワップ・アウトしていることがわかります。また、データベースの新規作成など、大量のリソースを使用するデータベース操作を実行すると、オペレータ・コンソールに SYSTEM-W-POOLEXPF エラーが表示されたり、INSFMEM エラーで操作が失敗することがあります。このような問題は Oracle Rdb 特有のものではなく、他にも、大量のリソースを使用するアプリケーションで同じ現象が発生することがあります。

この問題の原因として考えられるのは、OpenVMS Alpha バージョン 1.5 へのアップグレード手順の中で、仮想 I/O キャッシュのサイズが非常に大きな値に設定されてしまい、大量の物理メモリーが必要になるということです。

これが原因であるかどうかを判断するため、SYSGEN を使って、SYSGEN パラメータである VCC_FLAGS と VCC_MAXSIZE を確認してください。VCC_FLAGS パラメータは、仮想 I/O キャッシュを有効または無効にします。SYSGEN を使って、仮想 I/O キャッシュが有効になっているか確認してください。

```
SYSGEN>  SHO VCC_FLAGS
Parameter Name            Current    Default     Min.     Max.     Unit  Dynamic
--------------            -------    -------    -------  -------   ----  -------
VCC_FLAGS                       1          1         0        -1 Bitmask
```

VCC_FLAGS が 1 であれば、キャッシュは有効です（0 は無効）。キャッシュが有効になっていれば、VCC_MAXSIZE パラメータを確認してください。

```
SYSGEN>  SHO VCC_MAXSIZE
Parameter Name            Current    Default     Min.     Max.     Unit  Dynamic
--------------            -------    -------    -------  -------   ----  -------
VCC_MAXSIZE            2000000000       6400           02000000000 Blocks
```

デフォルトでは、6400 ディスク・ブロックのキャッシング用にメモリーが割り当てられています。これには、3.2MB のメモリーが必要です。現在 VCC_MAXSIZE の値に現在非常に大きな値が設定されている場合は、6400 ブロックを指定し、システムをリブートしてください。これで、問題は解消されます。◆

4.4.3 ワーキング・セット・クォータ・パラメータのチューニング

OpenVMS VAX OpenVMS Alpha

メモリー管理には、ワーキング・セット・クォータ・パラメータのチューニングが必要です。物理メモリー内の物理領域を必要とし、システム上で同時に実行されるプロセスの数と、使用するメモリー容量のバランスを維持することが、この作業の目的です。各プロセスに割り当てられている物理メモリー領域を、**ワーキング・セット**と呼びます。

一般的に、プロセスのワーキング・セットは、メモリー内にあるすべての有効ページで構成されています。通常、ワーキング・セットは、プロセスのページ・テーブル内にあるすべてのページのサブセットです。各プロセスのページは、ワーキング・セット、ページ・キャッシュ、ディスク上のいずれかに存在し、イメージ、セクション、ページングおよびスワッピングなどのファイルに格納されています。すべてのプロセスには、ワーキング・セットの上限ページ数の初期値として、WSDEFAULT がデフォルトで割り当てられます。ページ数が WSDEFAULT に達したら、各プロセスは、ワーキング・セット自動調整（Automatic Working Set Adjustment：AWSA）機能によって、ページ数をワーキング・セット・クォータ WSQUOTA まで拡張できます。プロセスは、使用可能なメモリー容量をチェックしなくても、WSDEFAULT から WSQUOTA まではページ数を増やすことができます。プロセスで、さらに多くのページが必要になった場合は、ワーキング・セットの最大値であるワーキング・セット・エクステント WSEXTENT まで増加できます。任意のプログラムに割当て可能なワーキング・セットの最大値は、必ず WSEXTENT を上回り、システム・パラメータの WSMAX として設定されます。

メモリー管理ストラテジは、各プロセスの WSQUOTA と WSEXTENT で設定されている値によって決まります。この値は、システム管理者が割り当てるユーザー認証ファイル（User Authorization File：UAF）、またはシステムが割り当てる DEFAULT レコードからプロセスに適用されます。WSDEFAULT、WSQUOTA および WSEXTENT を各ユーザーに割り当てるときには、注意が必要です。WSQUOTA には、十分に大きな値を設定し、各ユーザーのプロセスが、追加のページを要求する必要がないようにしてください。ただし、大きすぎると、メモリーに限りがある環境では、1 つのプロセスだけに不当にメモリーを割り当てることになるので、大きすぎない値を選択してください。一般的に、ワーキング・セットの最大値として適切な値は、パフォーマンスが急激に低下する直前の値です。したがって、ワーキング・セットの自動調整を考慮したストラテジに基づいて、各種プロセスのワーキング・セットの最大値の初期値を設定します。このようなストラテジを採用することにより、どのような場合にパラメータを調整できなくなるか、どのようにチューニングを実行すればよいかがわかります。自動的なワーキング・セット・クォータ・パラメータのチューニングには、一般に次の 2 通りストラテジがあります。このためのワーキング・セットの初期値設定に関する指針は、OpenVMS のマニュアルを参照してください。

- 常に変化する時分割環境で、作業負荷が大きなワーキング・セットを要求する場合にいつでも迅速に応答するためのチューニング
- 本番環境で、ワーキング・セットの緩やかな成長を要求する安定した変化の少ない応答時間を実現すめためのチューニング ◆

4.4.4 ユーザー・アカウント・パラメータの確認と設定

OpenVMS VAX OpenVMS Alpha

この項で説明する値は、最低値です。必要な設定値は、システムによって大きく変わる可能性があります。ここで説明している値は、Oracle Rdb のみに適用される値です。他の OpenVMS レイヤー製品で必要な値を Oracle Rdb で必要な値に追加し、必要に応じて各ユーザーの値を変更してください。

AUTHORIZE ユーティリティと UAF レコードの使い方は、OpenVMS のドキュメントを参照してください。Oracle Rdb ユーザーには、次の UAF パラメータが関連します。

- ENQLM
  ロック・キューの上限です。一度にキューに置けるロックの最大数です。

  Set high:2000

  レポジトリを使用している場合や、グローバル・バッファを有効にしている場合は、特殊な用途として、ENQLM に大きな値を設定する必要があることがあります。グローバル・バッファをデータベースで有効にしている場合、ユーザーの割当てセットの各ページにロックを 1 つ使用し、このロックは ENQLM にカウントされます。ENQLM は、プロセスが所有できるロックの数の上限です。

- WSDEFAULT、WSQUOTA、WSEXTENT
  デフォルトのワーキング・セット、ワーキング・セット・クォータ、ワーキング・セット・エクステント。
  WSDEFAULT は、ユーザーのプロセスのワーキング・セット・サイズの上限の初期値です。WSQUOTA は、指定した数の物理ページをユーザーに保証します。WSEXTENT は、システム上のあらゆるワーキング・セットの最大ページ数です。ワーキング・セット自動調整（Automatic Working Set Adjustment：AWSA）機能を有効にしている場合、これによって、ユーザーのワーキング・セットのサイズが決まります。

  Set high:WSDEFAULT 512、WSQUOTA 1024（使用可能なメモリー容量によって決まる）、WSEXTENT 2048 以上

  データベースで結合操作を何回も実行する場合は、結合要求を効率的に実行するために、さらに大きなページ・サイズが要求されます。データベースでグローバル・バッファを有効にし、グローバル・バッファが多数存在する場合は、WSDEFAULT、WSQUOTA および WSEXTENT の値を大きくします。自動的にワーキング・セット・クォータ・パラメータのチューニングでの、ワーキング・セットの初期値の設定については、OpenVMS のドキュメントを参照してください。
  投影操作を含む明示的なソート操作（SQL ORDER BY と DISTINCT）や、クエリー・オプティマイザによる暗黙的なソート操作では、クエリーに予想以上に時間がかかる場合や、ディスク・アクセスが大量に発生している場合は、WSEXTENT および WSQUOTA パラメータが適切に設定されているか確認してください。たとえば、WSEXTENT が 42,000 ページ、WSQUOTA が 20,000 ページの場合、WSEXTENT パラ

メータの値を WSQUOTA に近い値に設定します。つまり、
WSEXTENT=WSQUOTA+2000 の式に基づいて設定してください。ソート・ユーティリティ（SORT）（クエリーの実行では、何回もコールされます）は、この 2 つのパラメータの差（この場合は 2000 ページ）を使って、VM をスクラッチ領域に割り当てます。これによって、ページングが過剰に発生し、ディスク・ベースの一時ファイルの方が効率的です。
このような操作には、メモリー容量が少ない方が適切な処理ができます。システムの使用率が高くなると、OpenVMS オペレーティング・システムは、プロセスのワーキング・セット・エクステントにあるすべてのページを割り当てられなくなることがあります。使用率の高いシステムで、ページングが過剰に発生するのを防ぐために、ワーキング・セット・エクステントに小さな値を設定します。詳細は、DCL のヘルプを参照してください。

- FILLM
  オープン・ファイルの上限であり、ユーザーのプロセスが一度にオープンできるファイルの最大数を示します。

  Set high:OpenVMS V5.4 では 100、OpenVMS V5.5 では 100、OpenVMS V6.0 では 300

  ユーザーによるオープンが可能なファイルは、データベースの記憶領域、スナップショット・ファイル、.ruj ファイル、.aij ファイル、ルート・ファイル、ログ・ファイルなどです。アプリケーション固有のファイル（イメージやデータ・ファイル）や、レポジトリを使用したデータベース・アクセスも、オープン・ファイルの合計に追加してください。オープン・ファイルは、ユーザーのファイル数の上限にカウントされます。
  マルチファイル・データベースでは、システム上で実行している最大のアプリケーションについて、.rdb、.rda、.snp、.ruj、.aij および一時的な作業ファイルの合計に設定します。作成する .ruj ファイルの数が最も多いアプリケーションについて、.ruj ファイルの値をユーザー数（実際にはデータベースへの接続数）として使用します。この値を取得するには、RMU Dump Users コマンドを実行してください。
  FILLM クォータを超えると、現在の操作は強制終了し、超過クォータ・メッセージがユーザーに返されます。FILLM クォータを大きくする際は、新しい FILLM クォータが現在の CHANNELCNT よりも大きい場合、SYSGEN パラメータ CHANNELCNT も大きくする必要があります。

- BYTLM
  バッファ I/O バイトの上限であり、ユーザーのジョブが未処理のバッファ I/O 操作への転送として、一度に指定できる非ページングの動的システム・メモリーの最大バイト数です。

  Set high:20480 以上

- ASTLM
  非同期トラップの上限であり、プロセスでの未処理の AST の数の上限です。
  ASTLM の値は、次の式で計算してください。

```
ASTLM > DIOLM + 6 (or equals 24 if DIOLM is set to 18)
```

AST キューの上限は、一度に存在できる AST 操作と、処理待ちの起動要求の最大数です。ASTLM の値は、DIOLM の値に 6 を加算してください。DIOLM の値は、定義したデータベース・バッファの数以上に設定します。データベース・バッファは、データベースへ並行して書き戻されます。したがって、各バッファには、未処理 AST が存在する可能性もあります。
データベースのバッファ・パラメータの数の表示については、次に示す DIOLM の説明を参照してください。

- DIOLM
  ダイレクト I/O カウントの上限は、一度に未処理として存在できるダイレクト I/O 操作（通常はディスク）のカウント上限または最大数です。
  最初は 18 に設定します。
  パフォーマンスを向上するには、データベースでローカル・バッファを有効にしている場合、DIOLM の値は、定義したデータベース・バッファの数以上の値に設定します。データベースのバッファ・パラメータの値は、Performance Monitor の「Buffer Information」画面で確認できます。
  mf_personnel データベースにアクセスするプロセスについては、DIOLM には 20、ASTLM には 26 を設定すれば十分です。
  データベースでグローバル・バッファを有効にしている場合、DIOLM には、USER LIMIT パラメータ以上の値を設定してください（データベース内の任意のユーザーに設定可能な最大割当てセット）。RMU Show Users コマンドを実行すると、次の例のように、現行ノード上のデータベースの USER LIMIT パラメータのアクティブな値が表示されます。

```
$ RMU/SHOW USERS mf_personnel.rdb
   .
   .
- maximum global buffer count per user is 25
```

  mf_personnel データベースにアクセスし、グローバル・バッファを使用するプロセスについては、DIOLM には 25、ASTLM には 31 を設定すれば十分です。

- BIOLM
  バッファ I/O カウントの上限であり、一度に未処理として存在できるバッファ I/O 操作の最大数です。最初は 18 に設定します。
  BIOLM には、DIOLM と同じ値を設定します。DIOLM を変更した場合は、BIOLM も調整してください。

- PRCLM
  サブプロセス作成の上限であり、ユーザーのプロセスで一度に存在可能なサブプロセスの数の最大値です。
  この値は、1 に設定可能です。

- PGFLQUOTA
  ページング・ファイルの上限であり、ユーザーのプロセスがシステム・ページング・ファイルで使用できる最大ページ数です。
  これには、20,000 以上の値を設定してください。◆

5

クエリー・オプティマイザ

リレーショナル・データベース・モデルは、データベースに保管されているデータのユーザー・ビューを表しています。したがって、最も効率的な方法でデータを検索しようとすると、非常に複雑なタスクになってしまうこともあります。Oracle Rdb には、すべてのクエリーを自動的に分析して最も効率のよいデータ・アクセス方法を決定する**クエリー・オプティマイザ**（通常オプティマイザと呼ばれる）があります。

5.1 オプティマイザの担当機能

データベースのパフォーマンスは、ディスクに保存されている列または複数の列に I/O 操作を通してアクセスするデータベースの能力の影響を直接受けます。I/O 操作の回数が増えるにつれ、クエリーを満足する列の検索と取得に時間がかかるようになります。I/O 操作を最小に保つため、Oracle Rdb はクエリー・オプティマイザを使用します。オプティマイザの責任は、ユーザーが要求したデータを検索する最も I/O 効率のよい方法を決定することであり、さらにそのデータへのアクセス・パスを確立します。オプティマイザは、異なる取得方法を考慮しますが、どの検索ストラテジでも要求したデータと同じ結果が得られます。

最適化のほとんどの部分はクエリーのコンパイル中に実行されます。この**静的**な最適化は、クエリーの実行が行われる前にただ一度行われます。静的オプティマイザの一番の目的は、他の方法に比べて最も I/O 回数が少ないクエリーの実行ストラテジを実現することです。最高のストラテジに対する選択基準は、テーブルやインデックス・カーディナリティ、作業負荷やストレージ統計、任意の関係演算子が一般的に変更する入力データの量と分散の仮説など、要素の大まかな見積りを計算する純粋に数学的な最小実行コストにあります。これらの基本的な見積りと仮定は、効率的であることがわかりますが、場合によっては不正確で、クエリー・オプティマイザが最適でないストラテジを選択してしまうこともあります。

的オプティマイザの評価の誤りを補うため、Oracle Rdb クエリー・エグゼキュータは、クエリー検索プロセスを制御し、よりよいストラテジに動的に切り替えることができる**動的**なオプティマイザを使用します。現在のストラテジを選んだオプティマイザによって当初却下されたストラテジのコストが、クエリーの実行中によりコストの低い解決策を提供する点まで達したと見積られたとき、ストラテジ間で切替えが発生します。動的最適化は、見積りが正確ではないことを認識することにより、また間違いの発生を防止するばかりでなく発生した間違いを解決する強力なツール・セットを使用することにより、検索問題を解決する人的方法を模倣しています。

最も直接的でシンプルなクエリーの場合、静的オプティマイザは最も効率のよい実行計画を選択します。選択したストラテジが 1 つのテーブル検索レベルで最適でないことが判明したら、その方法を実行している間に動的オプティマイザが変更および修正できます。しかし、選択された解決策がまだ最適ではないと感じる場合があります。この章では、オプティマイザの概要について、別のアクセス方法でクエリーを最適化する方法およびオプティマイザに影響を与えるために何が行えるかを説明します。付録 C（C-1）では、アクセス方法の例を示しており、クエリーのコスト、オプティマイザ・ストラテジ、およびオプティマイザの実行を検証するため論理名 RDMS$DEBUG_FLAGS および構成パラメータ RDB_DEBUG_FLAGS の使用方法を説明します。

5.2 オプティマイザの用語

まず定義すべき Oracle Rdb クエリー・オプティマイザに特有の用語があります。5.2.1 項（5-3）から 5.2.3 項（5-3）でこれらの用語を定義します。他のクエリー・オプティマイザ用語は、導入時に定義されます。

5.2.1 述語の選択性

述語の選択性は、テーブルまたは中間結果テーブル内の列の総計に対する述語が真であるものの見積られた割合です。述語の選択性は、オペランド（つまり、クエリー内のデータまたは値）ではなく演算子のみに依存します。たとえば、オプティマイザは X > 10、X > 125 または Y > 70 の選択性が、3 つの場合すべてがオペレータ ">" を使用しているので、すべて 0.35 になると見積ります。述語の選択性は、できる限り速く列の総数を少なくするために、クエリーの最適化で可能な限り早く適用されます。

5.2.2 ストラテジ

ストラテジは、クエリーを解決して結果を生成するためにデータに対して実行される操作のシーケンスを指します。ストラテジはオプティマイザによって生成され、クエリーに対して最適なソリューションを提供します。最適なソリューションは、オプティマイザが見積ったものであり、必要なディスク I/O 回数は最小です。

5.2.3 コスト

コストとは、クエリー・ソリューションが必要と考えられるディスク I/O 操作の回数を見積ったものです。これは、最適なソリューション（最小コスト）を識別するためにオプティマイザが使用する物差しです。ソリューションのコストは、見積られたカーディナリティとオプティマイザが調査しているアクセス方法のタイプを基準にしています。コストに影響を与える要因には、次のものがあります。

- クエリーの構造
- クエリーの述語
- 結果の順序付けまたはグループ化
- 使用している検索のタイプ
- 使用している結合のタイプ（複数テーブルにアクセスする場合）
- 記憶領域のタイプ（同型または混合）
- 記憶領域のページ・サイズ（節 5.2.3.1（5-4）を参照）
- カーディナリティ統計
- 作業負荷統計（収集した場合）
- ストレージ統計（収集した場合）

論理名 RDMS$DEBUG_FLAGS または構成パラメータ RDB_DEBUG_FLAGS を使用してクエリーのコストを検証します。詳細は、付録 C（C-1）を参照してください。

5.2.3.1 オプティマイザが複数の記憶領域でデータを保管するテーブルのページ・サイズを見積る方法

Oracle Rdb を使用すると、テーブル内にある 1 つ以上の列の値を基準にして、テーブル列が保存される記憶領域を指定するストレージ・マップを定義できます。たとえば、次のストレージ・マップ定義では、テーブル TABLE1 内で列 COLUMN1 が 200 を超えない値を持つすべての列は、記憶領域 AREA1 に保存されます。TABLE1 のその他すべての列は記憶領域 AREA2 に保存されます。

```
SQL> CREATE STORAGE MAP MAP1 FOR TABLE1
  1> STORE USING (COLUMN1)
  2>    IN AREA1 WITH LIMIT OF ('200')
  3>    OTHERWISE IN AREA2;
```

2 つの記憶領域（AREA1 および AREA2）が異なるページ・サイズを持つことは可能です。テーブル用のストレージ・マップがこのテーブルに対する列を複数の記憶領域に保存するように指定するときはいつでも、オプティマイザは、取得にどの領域が使用されるかにかかわらず、そのテーブルに対するすべての見積りに最初の領域（ストレージ・マップ定義に表示されるように）のページ・サイズを使用します。

5.3 オプティマイザ統計

オプティマイザは、ソリューションのコストの決定に統計を使用します。オプティマイザに対してストラテジを選択するために次に示す 3 つのタイプの統計が利用できます。

- カーディナリティ
- 作業負荷
- ストレージ

Oracle Rdb は、自動的にカーディナリティ統計を維持します。作業負荷とストレージ統計値は RMU Collect Optimizer_Statistics コマンドを使用して収集される必要があります。

オプティマイザ統計の収集と維持についての情報は、『Oracle RMU Reference Manual』を参照してください。

5.3.1 カーディナリティ統計

カーディナリティ統計は、データ量の情報を獲得します。カーディナリティは量を参照します。カーディナリティ統計には、テーブル・カーディナリティ、インデックス・カーディナリティおよびインデックス接頭辞カーディナリティがあります。

5.3.1.1 テーブル・カーディナリティ

テーブル・カーディナリティは、テーブル内の列の数を明示します。各検索ソリューション（中間および最終の両方）にもカーディナリティがあります。完全なソリューションのカーディナリティは、クエリーが戻す列の数を見積ります。

テーブルのおよそのカーディナリティは、システム・テーブル RDB$RELATIONS のフィールド RDB$CARDINALITY に保管されます。これは、テーブルの 1 つの位置からおよそのカーディナリティを取得するすべてのデータベース・ユーザーに提供されます。

テーブルが初めてプロセスから参照されたとき、RDB$CARDINALITY の値はそのプロセスに関連したシンボル・テーブルにロードされます。これは、テーブルのカーディナリティがきわめて変わりやすい場合、または RDB$SYSTEM ストレージが読込み専用に設定されている場合、問題を起こす可能性があります。かなりの変更があった場合、Oracle Rdb には、各プロセスの RDB$CARDINALITY のコピーを更新する方法はありません。
RDB$CARDINALITY は、ユーザーがテーブルに列を追加したり削除するたびには更新されません。そのかわりに追加または削除された列数の統計は各ユーザーに対して保存されます。そして、コミット時に変更の総数が現在のカーディナリティの <log symbol> を超えると、RDB$CARDINALITY が更新されます。そうでなければ、プロセスが連結解除されたとき、RDB$CARDINALITY が更新されます。テーブルの列がさらに多くなると、RDB$CARDINALITY がオプティマイザ・ストラテジに影響を与えるほど異なることは少なくなります。

5.3.1.2 インデックス・カーディナリティ

テーブル・カーディナリティに加えて、Oracle Rdb は、データベース内で重複が許可されているソートされた各インデックスのカーディナリティの追跡を続けます。このようなインデックスのカーディナリティは、そのインデックスにおける個別値の数として定義されています。複数のセグメントを持つインデックスのサブセットではなく、完全なインデックス・カーディナリティのみが保存されます。

Oracle Rdb は、重複を許可する各インデックスに対してカーディナリティを維持するので、オプティマイザはインデックス内で発生する一意な値の件数を参照できます。その結果、オプティマイザはどれが一意な値の最高値を持つかについて、2 つ以上のインデックスの適合性に優先順位をつけることができます。

たとえば、SEX のインデックスがカーディナリティ 2 を持ち、テーブル内で 1000 列を持つ LAST_NAME のインデックスが 700 のカーディナリティを持つとします。オプティマイザは、これら 2 つのインデックスを区別でき、インデックス列の制限事項が同じであれば、より高いカーディナリティ値を持つほうを使用します。インデックス・カーディナリティ・メンテナンスは、インデックス付きの列を変更するトランザクションの更新、そして列を保存または削除するトランザクションにわずかな負荷を加えます。一意でないインデックスとは異なり、Oracle Rdb は、一意なインデックス・カーディナリティを維持しないので、一意なインデックスでは負荷が非常に低くなります。そのかわりにオプティマイザはインデックス

のコストを見積るために、一意なインデックスのカーディナリティと同じようにテーブル・カーディナリティを使用します。

インデックス・カーディナリティは、RDB$INDICES システム・テーブルの RDB$CARDINALITY フィールドに保存されます。

5.3.1.3 インデックス接頭辞カーディナリティ

インデックス接頭辞カーディナリティは、複数セグメントを持つソートされたインデックスの最初の部分の個別値キーの数です。つまり、最初のセグメントのみの個別値の数、最初のセグメントと 2 番目のセグメントを結合した個別値の数、最初、2 番目および 3 番目のセグメントを結合した個別値の数などを示します。この統計は、ソート・インデックスのみで意味があります。したがって、ハッシュ・インデックスでは維持されません。

オプティマイザは、スキャンが必要な B ツリー・インデックスの部分を決めるためにインデックス接頭辞カーディナリティを使用します。これによって、オプティマイザはインデックス検索のコストを見積ることができます。インデックス接頭辞カーディナリティは、テーブルからフェッチされた列の数を見積るためにも使用されます。

インデックス・カーディナリティは、RDB$INDEX_SEGMENTS システム・テーブルの RDB$CARDINALITY フィールドに保存されます。

5.3.1.4 テーブルおよびインデックス・カーディナリティの修正

システム・テーブルに保存された値が、テーブル列またはインデックス・キーの実際の数とは異なることも可能なので、オプティマイザは、廃止されたカーディナリティ値をベースにした検索ストラテジを作成できます。

RMU Collect Optimizer_Statistics コマンドを使用すると、カーディナリティを修正できます。たとえば、EMPLOYEES テーブルに対するテーブルとインデックス・カーディナリティを修正するには、次のコマンドを入力します。

```
$ RMU/COLLECT OPTIMIZER_STATISTICS mf_personnel/STATS=(CARDINALITY) -
_$ /TABLE=(EMPLOYEES)/LOG
$ rmu -collect optimizer_statistics mf_personnel ¥
> -statistics=¥(cardinality¥) -tables=¥(EMPLOYEES¥) -log
```

このコマンドは、複数のセグメントを持つソート・インデックスが EMPLOYEES テーブルで定義されている場合、インデックス接頭辞カーディナリティも更新することに注意してください。

一意なインデックスのカーディナリティは、RMU Collect Optimizer_Statistics コマンドを使用して変更される可能性があります。しかし、Oracle Rdb は、列が保存されたり、削除されたりしても、保存された値を使用しませんし、その値を更新しようともしません。

5.3.2 作業負荷統計

作業負荷統計は、重複および NULL ファクタの形式でデータ分散情報を獲得します。重複および NULL ファクタは、各作業負荷列グループに対して収集されます。作業負荷列グループには、テーブルからの 1 つ以上の列が含まれています。作業負荷列グループは、クエリー作業負荷をベースにオプティマイザによって識別されます。作業負荷列グループは、各作業負荷クエリーで指定された等価選択、等価結合、GROUP BY 句および DISTINCT 句から識別されます。各作業負荷列グループは、RDB$WORKLOAD システム・テーブルに保存されています。RDB$WORKLOAD における作業負荷列グループの収集は、**作業負荷プロファイル**として参照されます。

作業負荷統計を収集する前に、クエリー作業負荷プロファイルを生成する必要があります。SQL ALTER DATABASE 文または CREATE DATABASE 文の WORKLOAD COLLECTION 句を使用して、クエリー作業負荷プロファイルを生成します。次に、Oracle Rdb は RDB$WORKLOAD システム・テーブルを作成します。

すべてのあるいはほとんどのクエリー作業負荷がプロファイルされたら、作業負荷収集を無効にすることを考えてください。これは、通常の作業負荷の一部ではないその場限りのクエリーをベースにした新しい作業負荷列グループが偶然作成されないようにします。

作業負荷プロファイルが収集された後、RMU Collect Optimizer_Statistics コマンドを使用して各作業負荷列グループに対する重複および NULL ファクタを収集します。重複および NULL ファクタは、RDB$WORKLOAD システム・テーブルの対応する作業負荷列グループ・エントリに保存されます。

作業負荷統計は、オプティマイザを支援して、より正確にソリューション・コストおよびカーディナリティを見積ります。これは、一般的に作業負荷で種々のクエリーを最適に処理した結果得られます。

RDMS$DEBUG_FLAGS 論理名または RDB_DEBUG_FLAGS 構成パラメータが O と定義された場合、次の行はクエリーの最適化に対して作業負荷統計を使用することを示します。

```
Workload and Storage statistics used
```

5.3.2.1 列重複ファクタ

列重複ファクタは、作業負荷列グループにおける列の総数と個別値の数の比率です。作業負荷列グループの個別値あたりの重複する個数の算術平均です。これは、作業負荷列グループにおけるデータ分散に対するシングルポイント統計です。

この統計は、等価結合または集計のグループ化または投影操作の結果のカーディナリティと同様に等価選択の選択性ファクタの決定に使用されています。グループの場合、いくつのグループが形成されているかを知りたいでしょうし、プロジェクト（重複の除去）の場合、個別結果列の数を決定したいかもしれません。列または列のグループに等価選択を適応した場合、平均していくつの列が選択されるかを知っておく必要があります。列または列のグループに等価選択を適用した場合、結合した結果のカーディナリティを決定するには結合属性の fanout ファクタを知っておく必要があります。

列重複ファクタは、RDB$WORKLOAD システム・テーブルの RDB$DUPLICITY_FACTOR フィールドに保存されます。

この統計は、Oracle Rdb が自動的にメンテナンスするものではありません。RMU Collect Optimizer_Statistics コマンドを使用して、収集してください。

5.3.2.2 列 Null ファクタ

列 Null ファクタは、作業負荷列グループの少なくとも 1 つの列で NULL 値を持つ列の数とテーブル内の列の総数の比率です。等価結合および等価選択が NULL 値による列の削除を暗に含むので、列 NULL ファクタは、等価選択または等価結合操作に関係しない列の数の決定に使用されます。列 NULL ファクタは、外部結合の結果得られるカーディナリティの見積りにも使用されます。

列 NULL ファクタは、RDB$WORKLOAD システム・テーブルの RDB$NULL_FACTOR フィールドに保存されます。

この統計は、Oracle Rdb が自動的にメンテナンスするものではありません。RMU Collect Optimizer_Statistics コマンドを使用して、収集してください。

5.3.3 ストレージ統計

ストレージ統計は、インデックス・キー・クラスタ化ファクタ、インデックス・データ・クラスタ化ファクタおよびテーブル列クラスタ化ファクタなどのデータ・クラスタ化情報を獲得します。

RDMS$DEBUG_FLAGS 論理名または RDB_DEBUG_FLAGS 構成パラメータが O と定義された場合、次の行はクエリーを最適化するためにストレージ統計を使用することを示します。

```
Workload and Storage statistics used
```

5.3.3.1 インデックス・キー・クラスタ化ファクタ

インデックス・キー・クラスタ化ファクタは、それに関連したインデックス・キーおよび dbkey のフェッチに必要な I/O 操作の平均値です。これは、インデックス・キーと関連する dbkey がどれだけ近い位置にあるかを測定します。ソート・インデックスは、列のフェッチなしに全体スキャンまたは領域スキャンを行うために必要な I/O 操作の数を示しています。ハッシュ・インデックスは、列のフェッチなしにハッシュ・キーの検索を行うために必要な I/O 操作の数を示しています。この統計を使用すると、ソート・インデックスまたはハッシュ・インデックスを使用したインデックスのみの検索を行うために見積ったコストを改善できます。

インデックス・キー・クラスタ化ファクタは、RDB$INDICES システム・テーブルの RDB$KEY_CLUSTER_FACTOR フィールドに保存されます。

この統計は、Oracle Rdb が自動的にメンテナンスするものではありません。RMU Collect Optimizer_Statistics コマンドを使用して、収集してください。

5.3.3.2 インデックス・キー・クラスタ化ファクタ

インデックス・データー・クラスタ化ファクタは、1 つのインデックス・キーに関連したすべての列のフェッチに必要な I/O 操作の平均値です。これは、ソート・インデックス内のインデックス・キーの順序がデータ列の物理的配置とどれほどよく一致しているかを測定します。ハッシュ・インデックスでは、データへのアクセスが 1 度の I/O 操作で十分になるように、同じキー値を持つすべての列が同じ集合にどれほどうまく配置されているかを測定します。

インデックス・データ・クラスタ化ファクタは、インデックスが持つ隠しデータのクラスタ化を明らかにします。たとえば、すでにキー（列または列のグループ）でソートされているデータとともにロードされるテーブルがロードされるとします。ソート・インデックスがそのキーを使用して作成されている場合、そのインデックスは完全なまたは完全に近いデータ・クラスタ化になっています。テーブルがハッシュ・インデックスを通して列にロードされた場合、完全または完全に近いデータ・クラスタ化が発生します。

インデックス・データ・クラスタ化ファクタは、ソート・インデックス・スキャンまたはハッシュ・インデックス・ルックアップからデータをフェッチするコストを見積るために使用されます。完全なデータ・クラスタ化の場合、データをフェッチするコストは、ソート・インデックス・スキャンではデータ列の順次フェッチのコストに等しくなり、ハッシュ・インデックス・ルックアップでは 1 に等しくなります。

インデックス・データ・クラスタ化ファクタは、RDB$INDICES システム・テーブルの RDB$DATA_CLUSTER_FACTOR フィールドに保存されます。

この統計は、Oracle Rdb が自動的にメンテナンスするものではありません。RMU Collect Optimizer_Statistics コマンドを使用して、収集してください。

5.3.3.3 テーブル列クラスタ化ファクタ

テーブル列クラスタ化ファクタは、テーブルから 1 つずつ列を順にフェッチするために必要な I/O 操作の平均回数です。テーブルのデータ列が、記憶領域の別の集合にどれほどうまく保存されているかを測定します。この統計は、列の更新、または 1 ページにフィットできないのでいくつかの断片に分ける必要があった列により発生したデータ列の断片化による影響も取り込みます。この統計は、テーブルの順次スキャンによるコストの見積りに使用します。

テーブル列クラスタ化ファクタは、RDB$RELATIONS システム・テーブルの RDB$ROW_CLUSTER_FACTOR フィールドに保存されます。

この統計は、Oracle Rdb が自動的にメンテナンスするものではありません。RMU Collect Optimizer_Statistics コマンドを使用して、収集してください。

5.3.4 作業負荷およびストレージ統計の収集の制御

作業負荷やストレージ統計の使用には、いくつかのリスクがあります。次を行うことで、負の副作用の可能性を大きく減らすことができます。

- 個々のクエリーに対するクエリーのアウトラインを使用
- データベースからの作業負荷およびストレージ統計の削除
- RDMS$USE_OLD_COST_MODEL 論理名または RDB_USE_OLD_COST_MODEL 構成パラメータの定義

RDMS$USE_OLD_COST_MODEL または RDB_USE_OLD_COST_MODEL に任意の値を定義すると、オプティマイザは作業負荷またはストレージ統計を使用しません。RDMS$USE_OLD_COST_MODEL または RDB_USE_OLD_COST_MODEL を使用すると、次が行えます。

- ストレージおよび作業負荷統計を使用した場合としなかった場合でクエリーの作業負荷の最適化をテスト
- 特定のユーザー、プロセスおよびバッチ・ジョブに対する作業負荷およびストレージ統計の使用を選択的に無効にする

RDMS$USE_OLD_COST_MODEL 論理名または RDB_USE_OLD_COST_MODEL 構成パラメータが定義された場合、オプティマイザは、バージョン 7.0 以前に使用されていたコストとカーディナリティ機能を使用し、収集された作業負荷およびストレージ統計を無視します。

論理名または構成パラメータの割当てを解除して、オプティマイザの実行を可能にし、作業負荷およびストレージ統計を再度使用してコストおよびカーディナリティの見積りを行います。

5.4 クエリー・オプティマイザの概要

クエリーの複雑さによりますが、オプティマイザは、シングル・テーブル・アクセス方法、またはシングル・テーブル・アクセス方法と結合方法の組合せを使用して、そのクエリーが最小コスト・ソリューションに達するようにします。最小コスト・ソリューションを見つけるには、オプティマイザは異なる結合順序を使用して、ソリューションを作成し、コストが低いものを選択します。オプティマイザが行った結合順序の数は、結合したテーブルの数に直接関連しています。n 個のテーブルを結合すると、可能な結合順序は n!（n の階乗）個あります。たとえば、テーブルの 3 通りの結合が A、B および C とすると、オプティマイザは 3!（これは 6 になります）回の結合順序、つまり ABC、ACB、BAC、BCA、CAB、CBA を行います。

クエリーを解決するため、クエリー・オプティマイザは次のステップを実行します。

1. 可能なテーブルの検索方法を決定します。これは、クエリーで指定したテーブル列とテーブルで定義したインデックスをベースにしています。

2. テーブル・カーディナリティを見積ります。つまり、テーブルからアクセスする必要がある列の数を見積ります。この見積りは、テーブルで指定したクエリー述語をベースにしています。
3. 検索方法のコストを計算します。コストは、見積られたカーディナリティとオプティマイザが調査している検索方法のタイプをベースにしています。
4. 現在の検索方法とすでに生成されているその他可能性のあるソリューションを比較します。次に示すいずれかの条件が真であれば、新しい検索ソリューションを作成します。
   - 現在の検索ソリューションのコストが以前の検索ソリューションより低い場合
   - 現在の検索ソリューションが、あとでクエリー全体の解決に有用になる興味深い順序を生成する場合
5. クエリーが 2 つ以上のテーブルを指定している場合、現在のテーブルを既存の他のテーブルを含む中間ソリューションに結合する結合方法を決定します。
6. 次に示すいずれかの条件が真であれば、新しい結合ソリューションを作成します。
   - 現在の結合ソリューションのコストが以前の結合ソリューションより低い場合
   - 現在の結合が、あとでクエリー全体の解決に有用である興味深い順序を生成する場合
7. ステップ 6 が新しい結合ソリューションを作成した場合、ステップ 1 からステップ 6 を繰り返して、残りのテーブルのそれぞれを結合し、クエリーに対する完全なソリューションを生成します。
8. それぞれの結合順序に対してステップ 1 から 7 を繰り返します。最後に、最小コストのソリューションを選択することで完全なソリューションを選択します。

ステップ 4 からステップ 6 の間、以前のソリューションは、現在のものよりもコストかかかるか、何か興味深い順序を生成しなければ、取り除かれます。オプティマイザは、部分的なソリューションが既存の完全なソリューションよりもコストが高ければ、結合順序を使用した完全なソリューションを作成しません。最初に部分的なソリューションを取り除くと、クエリーの最適化に費やす時間の総計をきわめて低減することができます。

クエリーを作成する方法について過度に関心を持つ必要はありません。データベースが変更されたら、クエリー・オプティマイザは新しいアクセス・ストラテジを自動的に考案します。結合で指定した順序と選択した式での句の順序は、ほとんどの場合、オプティマイザが使用するクエリーを満足するための順序に影響しません。オプティマイザとの作業方法についての情報は、5.8 節（5-37）を参照してください。

5.5 シングル・テーブル検索方法

この項では、オプティマイザが使用できる 1 つのテーブルにアクセスするストラテジについて説明します。1 つのテーブルへのアクセスだけでなく、これらのストラテジは複数のテーブルの列にアクセスする結合の要素として使用できます。

RDMS$DEBUG_FLAGS 論理名または RDB_DEBUG_FLAGS 構成パラメータを使用して、クエリーに対する検索ストラテジを表示できます。付録 C（C-1）では、論理名および構成パラメータの使用方法を説明しています。また、選択された注釈付きのアクセス・ストラテジの例を提供しています。

クエリー・オプティマイザは、1 つのデータベースからデータを検索するために次の方法を使用できます。

- 順次検索
  テーブルの論理領域に対してデータベース・ページに順次アクセスし、クエリーで使用している選択式に関係なくテーブルのすべての列を読み込みます。
  RDMS$DEBUG_FLAGS または RDB_DEBUG_FLAGS を S と定義するとき、次の行は順次検索を示しています。

  ```
  Get   Retrieval sequentially of relation RELATION_NAME
  ```

- dbkey 検索
  dbkey（論理アドレス）列ポインタを通して直接テーブル・データにアクセスします。このアクセス・パスはどのインデックス・ノードにもアクセスしていないので、列それ自身だけがロックされています。RDMS$DEBUG_FLAGS または RDB_DEBUG_FLAGS を S と定義するとき、次の行は dbkey 検索を示しています。

  ```
  Get   Retrieval by DBK of relation RELATION_NAME
  ```

- インデックス検索
  特定のインデックス構造（ソートまたはハッシュ）にアクセスして、その列の dbkey を含むインデックス・キーを検索します。オプティマイザはデータ列をフェッチするために dbkey を使用します。このデータは、ソート・インデックスを使用している場合、インデックス順に渡されます。RDMS$DEBUG_FLAGS または RDB_DEBUG_FLAGS を S と定義するとき、次の行はインデックス検索を示しています。

  ```
  Get   Retrieval by index of relation RELATION_NAME
  Index name INDEX_NAME
  ```

- インデックスのみの検索
  インデックス・データのみにアクセスします。選択したインデックスには、クエリーで指定したすべてのテーブルの列が含まれています。したがって、これ以上列をフェッチする必要はありません。このオプティマイザは、ソート・インデックスを使用している場合、インデックス順にデータを渡します。RDMS$DEBUG_FLAGS または RDB_DEBUG_FLAGS を S と定義するとき、次の行はインデックスのみの検索を示しています。

  ```
  Index only retrieval of relation RELATION_NAME
  Index name INDEX_NAME
  ```

  インデックスのみの検索は、ソート・インデックスか、ハッシュ・インデックスで利用できます。

- OR インデックス検索
  クエリーでこれらのインデックスの述部が論理 OR で結び付けられている場合、1 つのテーブル上で定義された 2 つ以上のソート・インデックスまたはハッシュ・インデックスを使用します。RDMS$DEBUG_FLAGS または RDB_DEBUG_FLAGS を S と定義するとき、次の行は OR インデックス検索を示しています。

```
OR index retrieval
  Get  Retrieval by index of relation RELATION_NAME
    Index name INDEX_NAME_1 ...
  Conjunct  Get  Retrieval by index of relation RELATION_NAME
    Index name INDEX_NAME_2 ...
```

  2 つのインデックス、INDEX_NAME_1 および INDEX_NAME_2 は、通常のインデックス検索とは別に処理され、これらの検索でフェッチされた列は連結されます。オプティマイザは、異なる OR legs からフェッチされる列と同じ列が重複して発生するすれば破棄します。このデータは、特別な順序で渡されるのではありません。

- ダイナミック OR インデックス検索
  論理 OR で結合された 2 つ以上の述語を含むクエリーに対して、特定のソート・インデックスへアクセスします。データは、インデックス順に渡されます。RDMS$DEBUG_FLAGS または RDB_DEBUG_FLAGS を S と定義するとき、次の行はダイナミック OR インデックス検索を示しています。

```
Get  Retrieval by index of relation RELATION_NAME
Index name  INDEX_NAME [1:1...]2
```

  ダイナミック OR 最適化についての情報は、5.7.1 節（5-27）を参照してください。

- ダイナミック・リーフ検索
  インデックス間をダイナミックに（実行中に）選択します。RDMS$DEBUG_FLAGS または RDB_DEBUG_FLAGS を S と定義するとき、次の行はダイナミック・リーフ検索を示しています。

```
Leaf#01 ...
```

  ダイナミック・リーフの最適化についての情報は、5.7.2 節（5-31）を参照してください。

シングル・テーブルの場合、オプティマイザは、可能性がある検索ストラテジに対するコストを評価することで、最も安価な検索ソリューションを見つけます。ソリューションに対して見積った検索コストには、1 つ以上の利用可能なインデックスをスキャンするコストとデータ・レコードをフェッチするコストが含まれています。列をそれぞれ興味深い順序で生成する最も安価なアクセス・パスと順序付けられていない列を生成する最も安価なアクセス・パスが検査されます。

- クエリーにソートが必要ない場合には、オプティマイザは、順序付けられていない列を生成する最も安価なアクセス・パスを選択します。

- クエリーにソートが必要な場合、興味深い順序を生成するコストは、最も安価な順序付けられていないパスのコストに列を正しい順序でソートするコストを追加したものと比較され、最も安価なパスが選択されます。

インデックスを使用することで、特別な利点が得られる場合、オプティマイザはインデックス検索を選択します。しかし、クエリーが特別な出力順序を指定しなければ、インデックス列はこのテーブルに指定されたブール制限では使用されません。そして、与えられたインデックスは、データ領域から列をフェッチしなければ検索に使用できません。インデックスの選択に固有の理由はありません。この規則の唯一の例外は、クエリーが複合形式の記憶領域の列にアクセスする場合です。この場合の優先ストラテジは、すべての列を検索する必要がある場合でもインデックスを使用します。順次検索は記憶領域全体（領域内のすべてのテーブルのすべての列）を読み込む必要があるので、これは真です。

キー専用ブール最適化

キー専用ブール最適化はインデックス検索と関連して使用されます。可能な限り、オプティマイザはこの最適化を使用して列をフェッチする前にできるだけ多くのインデックス・キーをフィルタします。そしてキー専用ブール・オプティマイザは、フェッチする列の総数を減らすことでI/O 操作を節約することができます。

この最適化を説明するには、EMPLOYEES テーブルの FIRST_NAME 列と LAST_NAME 列を使用してソート・インデックス、EMP_FIRST_LAST が定義されていると仮定します。キー専用ブール最適化は、次のクエリーで使用されています。

```
SQL> SELECT EMPLOYEE_ID, FIRST_NAME, LAST_NAME
  1> FROM EMPLOYEES
  2> WHERE LAST_NAME STARTING WITH 'A'
  3> ORDER BY FIRST_NAME, LAST_NAME;
 EMPLOYEE_ID   FIRST_NAME   LAST_NAME
 00374         Leslie       Andriola
 00416         Louie        Ames
2 rows selected
```

RDMS$DEBUG_FLAGS 論理名または RDB_DEBUG_FLAGS 構成パラメータを S と定義すると、前のクエリーから次のストラテジが結果として得られます。

```
Conjunct        Get     Retrieval by index of relation EMPLOYEES
  Index name  EMP_FIRST_LAST [0:0] Bool
```

オプティマイザは、EMP_FIRST_LAST インデックスがクエリーで指定した順序を提供するので、これを使用します。したがって、その結果をソートする必要はありません。[0:0] 表記は、インデックス内のすべてのキーはスキャンされることを示しています。また、Bool 表記法は、キー専用ブールがこれらのインデックス・キーのそれぞれに適用されることを意味しています。ここで、オプティマイザは、キー専用ブールとして選択述語、LAST_NAME STARTING WITH 'A' を使用します。この述語 LAST_NAME STARTING WITH 'A' は、値が 'A' で始まるこれらのキーのみをスキャンするために使用することはできないことに注意してください。LAST_NAME は、先頭のセグメントではなく、最初のセグメントとは同等ではありません。

オプティマイザは、EMP_FIRST_LAST インデックス内のすべてのキーを順次スキャンします。キー専用ブールは、各インデックス・キーに適用され、不要なキーは除去されます。そして、オプティマイザは、キー専用ブールを満足するこれらの列のみをフェッチします。key-only 最適化なしで、EMPLOYEES テーブルのすべての列はフェッチされます。このタイプの最適化は、一般にかなりの数の I/O 操作を節約します。

オプティマイザは、次のクエリーでもキー専用ブール最適化を使用します。

```
SQL> SELECT EMPLOYEE_ID, FIRST_NAME, LAST_NAME
  1> FROM EMPLOYEES
  2> WHERE FIRST_NAME LIKE '%G%' IGNORE CASE;
Leaf#01 FFirst EMPLOYEES Card=100
  BgrNdx1 EMP_FIRST_LAST [0:0] Bool Fan=9
 EMPLOYEE_ID   FIRST_NAME   LAST_NAME
 00175         George       Siciliano
 00319         Glenn        Silver
 00202         Margaret     Harrington
 00179         Meg          Dallas
 00267         Roger        Saninocencio
 00209         Roger        Smith
6 rows selected
```

このクエリーの場合、オプティマイザは、キー専用ブールとして述語 FIRST_NAME LIKE '%G%' IGNORE CASE を使用します。すべての EMP_FIRST_LAST インデックスがスキャンされ、キー専用ブールが各インデックス・キーに適用されます。オプティマイザは、キー専用ブールを満足するこれらの列のみをフェッチします。EMPLOYEES テーブルの 100 列のうち、6 列のみが検索されます。

LIKE 述語は、この述語を満たすインデックス内のキーのみのスキャンには使用できませんが、そのかわり、インデックス内のすべてのキーがスキャンされる必要があることに注意してください。これは、CONTAINING 述語にも当てはまります。

最小 / 最大集計の最適化

オプティマイザは、クエリーが統計関数 MIN または MAX を含む場合、最小または最大集計の最適化を使用します。この最適化には、適切なソート・インデックスが必要であり、集計値はインデックスの一部である列に置かれる必要があります。この最適化に必要なその他の条件には、次の項目が含まれています。

- クエリーは、1 つの MIN 値または MAX 値を選択する必要があります。
- 値のベースになっている列は、インデックスの最初のセグメントであることが必要です。値が最初のセグメントのベースでない場合、すべての先行セグメントは、等価選択がベースであることが必要です。つまり先行セグメントをベースにした列はある値と等しい必要があるか、または、列は NULL であることが必要です。
- クエリー選択は、使用するインデックスに対して、1 つの範囲を指定する必要があります。

- 選択は、インデックスの外部の列をベースにしてはなりません。
- 最小 / 最大最適化に使用されるソート・インデックスは、そのセグメントの 1 つとして VARCHAR 列または COLLATING SEQUENCE 列を持ってはなりません。

オプティマイザは、昇順、降順、分割、または複数セグメントのソート・インデックスで最小 / 最大集計最適化を使用できます。この最適化が選択されたら、オプティマイザは、B ツリー・インデックス降下を実行して、最小列値または最大列値を位置付けます。このアプローチは、ある範囲のインデックス・キーをスキャンしたり、すべてのテーブル列を続けてスキャンして最小または最大列値を検索するよりも高速に行えます。

次のサンプル・クエリーは、最小 / 最大最適化を使用します。ストラテジ出力（RDMS$DEBUG_FLAGS または RDB_DEBUG_FLAGS を S と定義することから生成）は、最小 / 最大最適化を示す表記法を示しています。S フラグ表記法の詳細な解説は、付録 C（C-1）を参照してください。

```
SQL> SELECT MIN(EMPLOYEE_ID) FROM EMPLOYEES;
Aggregate       Index only retrieval of relation EMPLOYEES
  Index name  EMP_EMPLOYEE_ID [0:0]      Min key lookup
 00164
1 row selected
```

"Min key lookup" 表記は、最小 / 最大最適化を使用してオプティマイザが EMPLOYEE_ID 列の最小値を検索することを示しています。

```
SQL> SELECT MAX(EMPLOYEE_ID) FROM EMPLOYEES;
Aggregate       Index only retrieval of relation EMPLOYEES
  Index name  EMP_EMPLOYEE_ID [0:0]      Max key lookup
 00471
1 row selected
```

"Max key lookup" 表記は、最小 / 最大最適化を使用してオプティマイザが EMPLOYEE_ID 列の最大値を検索することを示しています。

```
SQL> SELECT MAX(EMPLOYEE_ID) FROM EMPLOYEES
  1>   WHERE EMPLOYEE_ID BETWEEN '00200' AND '00300';
Aggregate       Index only retrieval of relation EMPLOYEES
  Index name  EMP_EMPLOYEE_ID [1:1]      Max key lookup
 00287
1 row selected
```

この例では、オプティマイザは最小 / 最大最適化を使用して EMPLOYEE_ID 列の最大値を検索し、この最大値をインデックス範囲で検出しています（BETWEEN '00200' および '00300'）。「[1:1]」表記は、インデックス範囲の存在を示します。

次の例は、2 つのセグメントを持つインデックスを作成し、その後クエリーを使用しインデックスの 2 番目のセグメントをベースにして列から最大値を選択します。

```
SQL> CREATE INDEX SH_SALEND_SALARY
```

```
  1>   ON SALARY_HISTORY (SALARY_END, SALARY_AMOUNT);
SQL> --
SQL> -- Find the highest current salary.
SQL> SELECT MAX(SALARY_AMOUNT)
  1>   FROM SALARY_HISTORY
  2>   WHERE SALARY_END IS NULL;
Aggregate       Index only retrieval of relation SALARY_HISTORY
  Index name  SH_SALEND_SALARY [1:1]     Max key lookup
      93340.00
1 row selected
```

インデックス SH_SALEND_SALARY のキーには、SALARY_END および SALARY_AMOUNT の 2 つの列があります。オプティマイザは、最小 / 最大最適化を使用して、すべてが NULL の SALARY_END 値を持つインデックス・キーの範囲内で最大 SALARY_AMOUNT 値を検索します。

5.6 複数のテーブル・アクセス・ストラテジ

クエリーの解決（最適化）に対する複雑さの程度およびストラテジを生成するために必要な時間は、含まれるテーブルの数および各テーブルのインデックスの数によって決まります。この複雑さに対処するために、オプティマイザは完全なソリューションが作成されるまで、既存の中間ソリューションに 1 つずつテーブルを結合していきます。データ・ソース（レコード・ソース・ストリームまたは RSS）は、テーブル、サブクエリー・ソリューションまたはビューでもかまいません。

複数のデータ・ソースを結合するには、3 通りの異なる方法があります。

- クロス結合
- 一致結合
- マージ

これらのストラテジはネストでき、ネストの末端レベルでは、5.5 項（5-11）で説明しているように、1 つのテーブル・アクセス方法を使用します。クロスおよび一致ストラテジは、2 つ以上のデータ・ソースから列を結合しますが、マージ・ストラテジ（マージに対して使用）は 2 つ以上のデータ・ソースから列を連結します。

5.6.1 クロス結合

クロス結合方法は、ネスト結合とも呼ばれており、2 つ以上のデータ・ストリームを結合します。各データ・ストリームは、クロス結合表記法の**エントリ**に対応しています。クロス結合方法では、任意のエントリ中の列がソート順である必要はありません。
RDMS$DEBUG_FLAGS 論理名または RDB_DEBUG_FLAGS 構成パラメータを S と定義すると、クエリー・ストラテジ内の次の行は、クロス結合を示しています。

```
Cross block of 2 entries
   Cross block entry 1
   .
   .
   .
   Cross block entry 2
   .
   .
   .
```

次のステップは、クロス結合内で実行されます。

1. 列は、1 テーブルでも許可されるエントリ 1 のデータ・ストリームから検索されます。
2. 既存の制限述語がその列に適用されます。
3. その列が制限述語を満たしていれば、述語結合に一致するすべての列がエントリ 2 のデータ・ストリームから一度に一列ずつ検索されます。
4. ステップ 1 から 3 をエントリ 1 から検索された各列に対して繰り返します。

一般的に n エントリを持つクロス結合では、エントリ K からの処理された各列に対して、述語結合に一致するすべての列がエントリ K+1 から検索されます。そして、エントリ K+1 からの処理された各列に対して、述語結合に一致したすべての列が、エントリ K+2 から検索されます。以下同様です。

インデックス検索は、述語結合をベースにして一致するすべての列を検索する最も効果的な方法なので、エントリ 1 に続くエントリは、インデックス検索を使用します。オプティマイザが適切なインデックスを見つけた場合、エントリ 1 でもインデックス検索を使用できます。エントリ 2 はエントリ 1 の各列に対して処理されるため、最小に見積られたカーディナリティを持つデータ・ストリームはエントリ 1 に置かれます。したがって、後続のエントリの処理に対する繰返し回数は減少します。

内部結合に加えて、クロス結合アルゴリズムは、左の外部結合でも実行できます。右の外部結合は、2 つのオペランドを逆にしてそれを左の外部結合にすることで実行できます。左の外部結合を実行したとき、クロス結合アルゴリズムは、内部データ・ストリーム内でキーが一致する列が見つからなければすべての内部データ・ストリーム値を NULL に設定します。NULL 値を持つ外部データ・ストリームから左の外部結合の結果として一致しない列を返します。

5.6.2 一致結合

一致ストラテジには、2 つのバリエーションがあります。

- 一般的な一致のバリエーションに、一時テーブルを使用したソートまたはインデックス検索があります。

- 特別な一致のバリエーションは、ジグザグ一致と呼ばれます。これは、インデックス検索を行うときにインデックス・キーを使用して入力データ・ストリームをスキップすることで、特別な最適化を行います。

この項では、一般的な一致ストラテジについて説明します。5.6.3 項（5-21）では、ジグザグ一致結合について説明します。

RDMS$DEBUG_FLAGS 論理名または RDB_DEBUG_FLAGS 構成パラメータを S と定義すると、次の行は、一般的な一致検索ストラテジを示しています。

```
Match
  Outer Loop
    .
    .
    .
  Inner Loop
    .
    .
    .
```

一致ストラテジはクロス・ストラテジよりも明確です。これには、外部ループと内部ループの 2 つのデータ・ストリームがあります。このタイプの結合では、明示的またはインデックスを使用して、両方のループがソートされる必要があります。各ループの列は一度のみ処理されます。一致結合は、内部および外部ループの列が同じ昇順でソートされているときだけ動作します。言い換えると、降順インデックスなどは、一致結合方法では使用されません。一般的に一致結合はクロス結合よりもより効率的です。一致結合では、2 つのループのそれぞれを一度のみでスキャンしますが、クロス結合はエントリ 2（およびそれ以上のエントリ）を複数回スキャンします。

処理の最初に、オプティマイザは、内部ループおよび外部ループの両方から 1 列ずつ読み込みます。2 列の結合キーの値が異なる場合、両方のループでキーが等しくなるまで、オプティマイザは低いキーの値を持つループの列を読み続けます。同じキーの値が一致したとき、オプティマイザはこの一致したキーの繰返しを両方のループで検査します。そして、この等しいキーのグループに対して、5.6.1 項（5-17）で説明したようにクロス結合を実行します。オプティマイザは、一致するキーを検索する処理を繰り返します。

たとえば、TABLE1 および TABLE2 の 2 つのテーブルを考えてください。TABLE1 には、1、5、5、6 および 9 の値を返す 5 列があります。TABLE2 には、2、3、4、5、5 および 7 の値を返す 6 列があります。この一致は、例 5-1（5-19）に示すように 2 つのテーブルを処理します。

例 5-1 一般的な一致ストラテジ処理

```
    Outer Loop (TABLE1)              Inner Loop (TABLE2)
       **** Start search for a matching group ****
    Read first key (1)                Read first key (2)
                          1 < 2
           Switch to Outer Loop to use lower key value
```

```
   ************ Skip unmatched keys ************
Read next key (5)
 5 > 2 (no match)
 Switch to Inner Loop
                                  Read next key (3)
                                   5 > 3 (no match)
                                  Read next key (4)
                                   5 > 4 (no match)
                                  Read next key (5)
                                   5 = 5 (match)
   ********* Process equal key group *********
        Perform the nested loop algorithm
             as in Cross strategy
        Deliver pairs of matching records
Use last-read 5                   Use last-read 5
            Deliver pair of records
Use last-read 5                   Read next key (5)
            Deliver pair of records
                                  Read next key (7)
Read next key (5)
(still the same equal key group)
                                  Restart at first key 5
Use last-read 5                   Use restarted 5
            Deliver pair of records
Use last-read 5                   Read next key (5)
            Deliver pair of records
                                   Read next key (7)
Read next key (6)
(equal key group has ended)
   **** Start search for a matching group ****
Use last-read 6                   Use last-read 7
                     6 < 7
        Switch to Outer Loop to use lower key value
   ************ Skip unmatched keys ***********
Read next key (9)
 9 > 7 (no match)
 Switch to Inner Loop
                                  Read EndOfData mark
                                  Finish matching
```

オプティマイザは、一意な結合キーを持つデータ・ストリームを外部ループに設置しようとします。これにより、内部ループのデータ・ストリームを等価キー・グループの最初に戻す必要がなくなります。

内部結合に加えて、一致結合アルゴリズムは、左の外部結合および完全外部結合も実行できます。右の外部結合は、2 つのオペランドを逆にしてそれを左の外部結合にすることで実行できます。左の外部結合を実行したとき、一致結合アルゴリズムは、内部データ・ストリーム内でキーが一致する列がまったく見つからなければすべての内部データ・ストリーム値を

NULL に設定します。NULL 値を持つ外部データ・ストリームから左の外部結合の結果として一致しない列を返します。

完全外部結合を実行したとき、一致結合アルゴリズムは、外部データ・ストリーム内でキーが一致する列が見つからなければすべての外部データ・ストリーム値を NULL に設定します。これは、内部データ・ストリームから NULL 値とともに完全外部結合の結果の一部として、一致しなかった列を返します。

5.6.3 一致、ジグザグ

ジグザグ結合ストラテジは、一般的な一致方法のより高速なバリエーションです。この方法を使用する場合、内部ループ、外部ループまたは両方のループのデータ・ストリームはテーブルであることが必要であり、インデックス検索がそのテーブルで利用できることが必要です。ジグザグ・ストラテジは、内部 leg または外部 leg に適応できます。

RDMS$DEBUG_FLAGS 論理名または RDB_DEBUG_FLAGS 構成パラメータを S と定義する場合、次の行は、一般的なジグザグ一致検索ストラテジを示しています。

```
Match
  Outer Loop  (zig-zag)
    .
    .
    .
  Inner Loop  (zig-zag)
    .
    .
    .
```

ジグザグ処理は、5.6.2 項（5-18）で説明されている通常の一致ストラテジとは異なります。ジグザグ一致のバリエーションでは、オプティマイザは内部および外部ループ内にある複数のキーをスキップできます。通常の一致ストラテジは、内部ループ・キーを 1 つずつ読み込み、それを外部ループ・キーと比較して、一致しなかったものを内部ループ・キーから 1 つずつ取り除きます。

ジグザグ一致は、現在の外部ループ・キーを新しいポイントとして使用してインデックス・スキャンを再開することで、内部ループにある複数のキーをスキップできます。ジグザグ一致は、現在の内部ループ・キーを新しいポイントとして使用してインデックス・スキャンを再開することで、外部ループにある複数のキーをスキップすることもできます。非常に多くのキーをスキップできる場合、スキャン再開手続きは一番上のインデックス B ツリーから新しいポイントに迅速に降下することを選択するので、I/O 操作と CPU 時間を節約できます。いくつかのキーだけをスキップする必要がある場合、再開手続きは B ツリーを降下せずに、メモリー内のスキップを効果的に行い、CPU 時間を節約します。

例 5-2（5-22）は、例 5-1（5-19）のテーブルと同じテーブルを使用しています。これは、内部ループでのみ発生するジグザグ・スキップを示しています。外部ループはソートおよびスキャンされています。内部ループには、キー値へのアクセスに使用するインデックスがあります。

例 5-2 ジグザグ一致ストラテジの処理

```
Outer Loop (TABLE1)              Inner Loop (TABLE2)
                                 Zigzag side
   **** Start search for a matching group ****
Read first key (1)                Read first key (2)
                       1 < 2
        Switch to Outer Loop to use lower key value
   ************ Skip unmatched keys ************
Read next key (5)
 5 > 2 (no match)
 Switch to Inner Loop
                                  Skip to next key >=5
                                  Skips keys 3,4
                                   Gets key 5
                                   5 = 5 (match)
   ********* Process equal key group *********
   Processing continues as in a regular match strategy
```

Oracle Rdb は、外部ループのデータ・ストリームがテーブルであり、インデックス検索が使用されている場合、外部ループにジグザグ・キー・スキップをサポートします。

通常の一致結合に対するジグザグ一致結合の利点は、内部、外部または両方のループのインデックスを使用してスキップし、より高いキーに移動できる点にあります。

5.6.4 マージ

マージ・ストラテジは、2 つ以上のデータ・ソースのデータを連結します。このストラテジは、オプティマイザが UNION オペレータを発見したときに実行します。

RDMS$DEBUG_FLAGS 論理名または RDB_DEBUG_FLAGS 構成パラメータを S と定義する場合、次の行は、一般的なマージ検索ストラテジを示しています。

```
Merge block of n entries
   Merge block entry 1
      ...
   Merge block entry 2
      ...
   .
   .
   .
```

結合操作とは異なり、マージ・ストラテジは複数のデータ・ソースで共通のデータをソース内にただ 1 つに含まれるデータと同じように返すことができます。マージ・ストラテジでは、一致機能ではなく追加機能を使用して 2 つ以上のデータ・ソースを結合します。

マージ・ストラテジは、次の一般的なステップに従います。

1. entry1 は、エントリ本体に記述されているストラテジに従って、完全に処理されます。

2. entry2 は、エントリ本体に記述されているストラテジに従って完全に処理され、その結果は entry1 の作業の後に追加されます。

5.6.5 結合順序

クエリー・オプティマイザの主要な機能の 1 つは、最も効率的な結合順序を識別するためテーブルが互いに結合できる可能な方法をすべて考慮することです。しかし、オプティマイザが考慮する可能性のあるソリューションの範囲に次の制限を使用できます。

- 外部結合
- 派生テーブル
- 表示

内部結合や完全な外部結合とは異なり、左の外部結合と右の外部結合は非可換操作です。つまり、左または右の外部結合オペランドは、逆にすることができません。たとえば、結合式 A left outer join B は、B left outer join A とは異なる結果が得られます。

また、内部結合とは異なり、外部結合（左、右または全）は結合性のない操作です。これは外部結合のオペランドは交換できないことを意味しています。たとえば、結合式 (A inner join B) inner join C は、A inner join (B inner join C) と同じ結果をもたらします。ところが、結合式 (A full outer join B) full outer join C は A full outer join (B full outer join C) と異なる結果をもたらします。

次の項では、結合順序について、さらに詳しく説明します。指定していなければ、"outer join" が左、右または完全外部結合を参照することに注意してください。次の例では、シンボル IJ、LJ、RJ および FJ は、それぞれ内部結合、左外部結合、右外部結合および完全外部結合を示すために使用されることにも注意してください。

5.6.5.1 結合演算子の順序

連続する外部結合では、SQL はデフォルトの結合順序を左から右に定義します。しかし、カッコを使用すれば、デフォルトを変更することができます。たとえば、結合式 A LJ B LJ C を考えてみてください。

カッコがなければ、SQL はデフォルトの結合順序 [A LJ B] LJ C で、次のように ON 句または USING 句を加える必要があります。

```
A L J B ON .... L J C ON ....
```

カッコを使用すると、次のように結合順序を変更できます。

```
A L J (B L J C ON ....) ON .....
```

2 番目の式は、最初の式と同じ結果にならないことに注意してください。これらの 2 式では、意味が異なるためです。SQL に対して、両方のテーブルにある同じ名前の列を結合に使用す

ることを効率的に指定できる NATURAL 修飾子を使用すると、ON 句または USING 句は必要ないことにも注意してください。

5.6.5.2 結合オペランドの順序

左または右の外部結合のオペランドは非可換な操作です。つまり、このオペランドは逆にはできません。たとえば、A LJ B は B LJ A と同じではありません。しかし、A LJ B は B RJ A と同じ結果が得られます。この性質は、オプティマイザが正しい結果を保証しながら、すべての右の外部結合を左の外部結合に変換する場合に使用します。この変換が行われるので、結合アルゴリズムは内部、左外部および完全外部結合のみを実行すればよくなります。

内部結合と完全外部結合は、可換性であるので、これらのオペランドは逆にできます。オプティマイザはオペランドを交換してソリューションを試し、最もコストがかからないソリューションを見つけ出します。

5.6.5.3 結合オペレータの組合せ

外部結合の処理を意味的に正しく行うため、外部結合のオペランドは変更できません。したがって、外部結合オペランドは他の結合のオペランドと交換することができません。

たとえば、次の結合式を考えてください。

```
A IJ B ON ... IJ C ON ....
```

内部結合の場合、オプティマイザは任意の順序で A、B および C を結合します。明白な結合条件が A と C の間になくても最初に A と C を結合できます。そのように行うと正しい結果が得られ、最高のパフォーマンスが提供できます。

比較のため、次の結合式を考慮してください。

```
A L J B ON ... L J C ON ....
```

この問題は最初にオペランド A と B を結合した後で、オペランド C を結合することのみによって解決できます。

内部結合および外部結合を混合する場合、外部結合は、結合順序になんらかの制限を加えます。各外部結合に対する 2 つのオペランドは、常に互いに結合します。そして、これらのオペランドは他のオペランドと交換することはできません。

たとえば、次の結合式を考えてください。

```
A L J B IJ C
```

可能な結合順序は、2 通りのみです。

```
[A L J B] IJ C
C IJ [A L J B]
```

上記の結合式で LJ が FJ に代わると次のようになります。

```
A FJ B IJ C
```

そして、4 通りの可能な結合順序があります。

```
[A FJ B] IJ C
[B FJ A] IJ C
C IJ [A FJ B]
C IJ [B FJ A]
```

この例では、内部結合のオペランドは、完全な外部結合のオペランドと同様に交換できます。

5.6.5.4 結合順序の変更

集計も行うのでなければ、派生テーブル構造体を使用して結合順序を変更する必要はありません。したがって、結合順序を変更するには、次の結合式のように丸カッコのみを使用することをお薦めします。

```
A LJ (B LJ C ON ....) ON ....
```

前の例は、次の結合式よりも優れた構造です。

```
A L J (B L J C ON ....) as DTAB ON ....
```

後の結合式は、オプティマイザが効率のよいストラテジを生成することを防げます。

カッコの使用は、外部結合セマンティックスがそのような制限を要求しなければ、結合順序におけるあらゆる制限を強いるものではないことに注意してください。たとえば、次の 3 通りの結合式は等価です。

```
(A IJ B) IJ C
A IJ (B IJ C)
A IJ B IJ C
```

しかし、次の 2 つの結合式は等価ではありません。

```
(A IJ B) L J C
A IJ (B L J C)
```

この場合、結合順序の制限はカッコの使用よりも左の外部結合の存在によって、強制されます。

派生テーブルとビューを使用すると、結合の順序に制限を加えます。たとえば、次の結合式はオプティマイザに対して A と B を常にいっしょに結合させます。したがって、オペランド A および B は、オペランド C と交換することは許されません。

```
(A IJ B) AS DTAB IJ C
```

次のクエリーはビューを含んでおり、オプティマイザに A と B を常にいっしょに結合させるように強制します。

```
CREATE VIEW V AS SELECT * FROM A NATURAL JOIN B;
SELECT * FROM V NATURAL JOIN C;
```

5.6.5.5 派生テーブルの使用

SQL を使用すると、選択リスト、WHERE 句および FROM 句など様々な領域にサブクエリーを含むクエリーを記述できます。これらのサブクエリーはネストできます。

FROM 句のサブクエリーは、**派生テーブル**と呼ばれています。派生テーブルの例を次に示します。

```
SELECT ... FROM
  (SELECT .... FROM .... WHERE ...) AS T1,
  (SELECT .... FROM
       (SELECT .... FROM .... WHERE ...) AS T3,
       (SELECT .... FROM .... WHERE ...) AS T4
   WHERE T3.x = T4.x ) AS T2
WHERE T1.y = T2.y;
```

派生テーブルを使用すると、二重集計や結合集計などの複雑なオペランドを簡単な式で表すことができます。しかし、派生テーブルを使用して集計なしで選択と結合のみを実行しても、オプティマイザが利用できる選択肢が制限されているため、結果としてパフォーマンスは低くなります。

クエリーが集計を含んでいない場合、派生テーブルを使用する理由はありません。**クエリー・フラット化**と呼ばれるプロセスを使用して、サブクエリーを親クエリーに戻すのが効率のよい方法です。

たとえば、次のクエリーは非効率的です。

```
SELECT E.EMPLOYEE_ID, E.LAST_NAME, J.JOB_CODE
 FROM
  (SELECT EMPLOYEE_ID, LAST_NAME
     FROM EMPLOYEES
    WHERE STATE = 'MA') E,
  (SELECT EMPLOYEE_ID, JOB_CODE
     FROM JOB_HISTORY
    WHERE JOB_END IS NULL) J
WHERE E.EMPLOYEE_ID = J.EMPLOYEE_ID;
```

このクエリーは、次のように表現した方が優れています。

```
SELECT E.EMPLOYEE_ID, E.LAST_NAME, J.JOB_CODE
  FROM EMPLOYEES E, JOB_HISTORY J
 WHERE E.EMPLOYEE_ID = J.EMPLOYEE_ID
   AND J.JOB_END IS NULL
   AND E.STATE = 'MA';
```

前者のクエリーの処理時には、オプティマイザは各派生テーブルを別の存在として考え、それぞれに対するクエリーのサブプランを提出します。そして 2 つのサブプランの結果を組み合せて、最終的な結果を出します。これは、オプティマイザが 1 つの広域的に最適化されたクエリープランを生成できる 2 番目のクエリーと比較して、効率的に大きく劣る方法です。

5.7 動的最適化

現行バージョンの Oracle Rdb では、1 つのテーブルを検索する場合のみ動的最適化が使用されます。言い換えると、これはリーフレベルの動的最適化です。動的最適化が行われている間は、異なるテーブルに対するアクセス・ストラテジが同時に実行され、各ストラテジは、クエリーを満足する列を生成します。ある時間が経過した後に最高速で動作すると思われるストラテジが選択され、列の大半を実行します。その一方で、静的最適化は、クエリーのコンパイル時にテーブルへのアクセス・ストラテジを選択し、実行時間の調整を動的最適化に残します。

静的オプティマイザは、最高の検索ストラテジを選択するときに不可避な間違いを犯すことがあります。間違いの可能性の 1 つとして、順次検索とインデックス検索を間違えて選択することがあります。間違いの可能性の 2 つめは、非常に短いインデックス範囲を利用できるときに、長いインデックス・スキャンを選択してしまう場合があります。動的最適化は、正しいインデックスやインデックスの組合せを使用することで、従来の最適化技術を改良しています。

任意の数のテーブルを動的に最適化できますが、結合やマージの場合は静的オプティマイザが各分岐を 1 つのテーブルの実行ツリーに変換しています。個々のテーブルへのアクセスは、5.5 項（5-11）で説明する従来のストラテジの 1 つを使用するか、5.7.2 項（5-31）で説明する動的リーフレベル・ストラテジの 1 つを使用して行います。

クエリー・オプティマイザには、次に示す 2 つの動的最適化があります。

- 動的 OR 最適化
- 動的リーフレベル最適化

5.7.1 動的 OR 最適化

次に示す条件がある場合、オプティマイザは、動的 OR 最適化ストラテジを選択します。

- ソート・インデックスまたはハッシュ・インデックスが利用可能
- 2 つ以上のインデックス範囲を定義した次のオペレータのうちの 1 つがクエリーに含まれている
  - IN
  - OR

動的 OR 最適化は、従来の OR インデックス検索（5.5 項（5-11）参照）に対して次の利点を提供しています。

- 各範囲に対して 1 対の NDX と GET ブロックを使用するかわりに、唯一の NDX と GET ブロックのペアを使用します。各範囲に対してインデックスのオープンとクローズを行うかわりに、一度だけオープンとクローズを行います。

- キー範囲が最小のところでインデックス範囲のソートを行い（範囲述語にホスト変数が含まれている場合、クエリーのコンパイル時か、クエリーの実行時）、その後、重なっている範囲を結合して余分なキーのスキャンを除きます。
- インデックス・キーの順序で結果を引き渡します。このタイプの最適化では、クエリーによって明示的な順序が指定されている場合、集計用データのグループ化やクエリーによって複製の除去が指定されている場合、またクエリー検索が一致結合の一部として処理されている場合に、ソートを使用する必要がなくなります。次のケースでは、動的 OR 最適化によって、結果をインデックス・キー順序で返すためのソートが不要になる場合を説明します。
  - 次のクエリーは、明示的な順序で列を返す必要があるクエリーに対して、ソートの使用を回避する動的 OR 最適化を示しています。

```
SELECT EMPLOYEE_ID, FIRST_NAME, LAST_NAME
   FROM EMPLOYEES
   WHERE EMPLOYEE_ID IN ('00345', '00166', '00222')
   ORDER BY EMPLOYEE_ID;
Conjunct        Get     Retrieval by index of relation EMPLOYEES
  Index name  EMPLOYEES_HASH [1:1...]3
 EMPLOYEE_ID   FIRST_NAME   LAST_NAME
 00166         Rick         Dietrich
 00222         Norman       Lasch
 00345         James        Stornelli
3 rows selected
```

    従来の静的 OR 最適化では、この結果はインデックス・キー順になりません。したがって、要求された EMPLOYEE_ID 順でクエリーの結果を引き渡すためにはソートが必要になります。

  - 次のクエリーは、動的 OR 最適化の使用について説明しています。これは、あるクエリーが集計組込み関数を使用して値を計算するために、データのグループ化が必要な場合、ソートの使用を回避します。

```
SELECT E.EMPLOYEE_ID, AVG(SALARY_AMOUNT) AS AVG_SALARY
   FROM EMPLOYEES E, SALARY_HISTORY SH
   WHERE E.EMPLOYEE_ID = SH.EMPLOYEE_ID AND
         E.EMPLOYEE_ID IN ('00345', '00166', '00222')
   GROUP BY E.EMPLOYEE_ID;
Aggregate       Conjunct
Match
  Outer loop
    Conjunct        Index only retrieval of relation EMPLOYEES
      Index name  EMP_EMPLOYEE_ID [1:1...]3
  Inner loop      (zig-zag)
    Conjunct        Get     Retrieval by index of relation SALARY_HISTORY
      Index name  SH_EMPLOYEE_ID [1:1...]3
 E.EMPLOYEE_ID                AVG_SALARY
```

```
        00166            1.691766666666667E+004
        00222            1.307625000000000E+004
        00345            5.160700000000000E+004
      3 rows selected
```

ソート・インデックスで動的 OR 最適化を説明するため、ソート・インデックス DEG_COLLEGE_CODE が DEGREES テーブルの COLLEGE_CODE 列で定義されているとします。次の例では、（COLLEGE_CODE ='USCA' OR COLLEGE_CODE = 'FLU'）に対する簡略表記法である IN オペレータを使用したクエリーを示します。クエリーに続いて、RDMS$DEBUG_FLAGS 論理名または RDB_DEBUG_FLAGS 構成パラメータが S と定義されている場合に出力されるストラテジの結果が表示されています。

```
SQL> SELECT * FROM DEGREES
  1>   WHERE COLLEGE_CODE IN ('U

SCA', 'FLU');
Leaf#01 FFirst DEGREES Card=165
  BgrNdx1 DEG_EMP_ID [0:0] Fan=17
  BgrNdx2 DEG_COLLEGE_CODE [1:1...]2 Fan=17
 EMPLOYEE_ID   COLLEGE_CODE   YEAR_GIVEN   DEGREE   DEGREE_FIELD
 00187         FLU                  1966   BA       Arts
 00206         FLU                  1979   BA       Arts
 00231         FLU                  1967   BA       Arts
 00249         FLU                  1977   BA       Arts
 00415         FLU                  1976   MA       Applied Math
 00176         USCA                 1982   MA       Applied Math
 00198         USCA                 1974   BA       Arts
 00217         USCA                 1974   BA       Arts
8 rows selected
SQL>
```

例で使用されている表記法の詳細は、付録 C（C-1）を参照してください。ただし、"BgrNdx2" で始まる出力の 3 行目は、次のように説明されます。

- DEG_COLLEGE_CODE は、データ検索に使用するインデックスを示しています。
- [1:1...]2 はオプティマイザが動的 OR インデックス検索を使用していることを示しています。

  - 最初の「1」は、最初の範囲（COLLEGE_CODE = 'USCA'）のセグメントに対する下限を示しています。
  - 2 番目の「1」は、最初の範囲（COLLEGE_CODE = 'USCA'）のセグメントに対する上限を示しています。
  - 「...」表記法は、その他のインデックスの範囲（この場合、もう 1 つの範囲である COLLEGE_CODE = 'FLU'）を示しています。
  - 最後の「2」は、重複している範囲を結合する前のインデックス範囲の総数を指定します。

オプティマイザが前のクエリーを実行する場合、次の各ステップが実行されます。

1. DEG_COLLEGE_CODE インデックスがオープンされて B ツリー降下が実行され、最初のインデックス範囲の下限キーが位置付けられます。
2. 次を取得する操作を実行してインデックスからキーをフェッチします。
3. ブール・チェック（最初の範囲であれば、COLLEGE_CODE = 'USCA' で、2 番目の範囲であれば COLLEGE_CODE = 'FLU'）が行われ、キーが現在の範囲に属しているかどうかを決定します。
4. そうであれば、列をフェッチして渡します。ステップ 2 から 4 は、ステップ 3 のブール・チェックが失敗するまで繰り返されます。さもなくば、次の範囲の下限キーを使用してキースキップを実行し、次の範囲の最初の開始位置を得ます。そして、ステップ 2 から 4 を繰り返して、新しい範囲からキーをスキャンします。
5. すべてのインデックス範囲をスキャンしたら、インデックスをクローズします。

キースキップ・メカニズムは最適化されているので、範囲間のギャップのサイズによって、次の範囲の最初に位置付けるために B ツリー降下を実行するかしないかが決まります。

2 セグメント・インデックスを使用したその他の例を考えてみましょう。ソート・インデックス DEG_COLLEGE_CODE_YEAR_GIVEN が、DEGREES テーブルの COLLEGE_CODE 列および YEAR_GIVEN 列にあると仮定します。動的 OR 最適化は次のクエリーで使用されます。

```
SQL> SELECT * FROM DEGREES
  1> WHERE COLLEGE_CODE = 'STAN' AND
  2>       (YEAR_GIVEN = 1973 OR YEAR_GIVEN = 1981);
Leaf#01 FFirst DEGREES Card=165
  BgrNdx1 DEG_COLLEGE_CODE_YEAR_GIVEN [2:2...]2 Fan=17
EMPLOYEE_ID   COLLEGE_CODE   YEAR_GIVEN   DEGREE   DEGREE_FIELD
00185         STAN           1973         MA       Elect. Engrg.
00201         STAN           1973         BA       Arts
00208         STAN           1981         BA       Arts
00226         STAN           1981         BA       Arts
00230         STAN           1981         MA       Statistics
00374         STAN           1981         PhD      Statistics
6 rows selected
```

クエリー内で 2 つのインデックス範囲が指定されます。

- 最初の範囲は、COLLEGE_CODE = 'STAN' と YEAR_GIVEN = 1973 です。
- 2 番目の範囲は、COLLEGE_CODE = 'STAN' と YEAR_GIVEN = 1981 です。

表記法 [2:2...]2 は、オプティマイザが動的 OR インデックス検索を使用していることを示します。最初の範囲に 2 つのセグメントに対する下限と 2 つのセグメントに対する上限があり、2 つのインデックス範囲の総計があります。

5.7.2 動的リーフ最適化

動的リーフ最適化（DynamOpt とも呼ばれる）は、利用可能なインデックスが複数ある場合、最高の組合せのインデックスを選択します。インデックス・スキャンは、クエリーの結果得られる非常に正確なデータ数を与えます。スキャンの結果見積られた選択性は、利用可能なインデックスをスキャンのコストの順番で並べ替え、最高のインデックスを決定するために使用されます。

ある範囲で少なくとも 1 つのインデックスまたは等価制限が利用可能な場合、オプティマイザは、動的リーフ・ストラテジを自動的に選択し、クエリーがテーブルに対してどのような更新も行いません。動的リーフ最適化の実験に基づく負荷を避けるため、4 つのケースでオプティマイザは従来のインデックス検索ストラテジを選択します。次に 4 つの例外を示します。

- 正確にキーが一致する一意なインデックス（つまり、すべてのセグメントが等価であると指定されています。）
- 有用なインデックスが他にない場合のインデックスのみの検索
- 要求した順序で列を生じるインデックスのみの検索（他にも有用なインデックスがあるかもしれませんが、無視されます）
- 2 つ以上のインデックスを使用してテーブルからデータにアクセスする OR インデックス検索

動的リーフ・ストラテジは、1 つのテーブル検索に対する一番主要な要求とインデックス構成を満たします。4 つの動的リーフレベル検索ストラテジには、次のものがあります。

- バックグラウンドのみ（RDMS$DEBUG_FLAGS または RDB_DEBUG_FLAGS ストラテジ・ディスプレイに「BgrOnly」と表示されます）。これはすべての検索時間を最適化します。このストラテジは、5.7.2.1 項（5-34）で説明しています。
- Fast First（「FFirst」）。これは、クエリーの最初の数列に対する検索時間を最適化します。このストラテジは、5.7.2.2 項（5-35）で説明しています。
- Index Only（「NdxOnly」）。これは、クエリーが要求したすべての列にインデックスが含まれている場合、データをフェッチする必要がある有用なインデックスが 1 つ以上存在している場合にオプティマイザが使用します。このストラテジは、5.7.2.3 項（5-36）で説明しています。
- Sorted（「Sorted」）。これは、既存のインデックスが要求したソート順序を提供し、1 つ以上の有用なインデックスが存在する場合に、オプティマイザが使用します。このストラテジは、5.7.2.4 項（5-37）で説明しています。

4 つのストラテジのすべては、2 つの主要な原理によって決定されます。

- ストラテジは互いに競争関係にあります。選ばれた勝者は列を作成します。
- 競争を行っている間、利用価値の高いデータは各インデックス・スキャンによって蓄積されているので、その他のスキャンやフェッチがより効率的に行われます。

動的リーフレベル最適化の一番の利点は、同じクエリー内で実行中であっても、次の各項目にかかわらず、テーブル・アクセスの各インスタンスに対して最適に近いパフォーマンスを提供できることです。

- データ配分
- 列の相関性
- 0 または少数の列を選択して、1 つのインスタンスに渡す場合
- すべての列または多くの列が同じリーフ実行内の他のインスタンスに渡された場合

バックグラウンド・プロセスおよびフォアグラウンド・プロセスという 2 つの動的最適化コンテキストがあります。この 2 つのプロセスは、ユーザー・プログラムの 1 つのスレッド内か、対話型 SQL 内のどちらかに実装されています。個々のスキャンは、列に対して同じ相対速度を維持しながら、入れ替えて実行できます。

バックグラウンド・プロセス

バックグラウンド・プロセスは、1 つ以上のインデックスの範囲をスキャンし、dbkey のソート・リストおよびメモリー内でソートした dbkey リストか、メモリー内のビットマップとして表されたフィルタを引き渡します。フィルタは、フォアグラウンド・プロセスが使用して、データ領域から不要な列をフェッチしないようにします。ソートされた dbkey リストは、最終的な列のフェッチと引渡しに使用する最終ステージで使用されます。引渡しのために dbkey を事前ソートしておくことは、インデックスを使用して配置されていないデータをフェッチするクエリーの効率を改善します。バックグラウンド・プロセスは、列をユーザーに渡せません。

バックグラウンド・プロセスは、各バックグラウンド・インデックス（BgrNdx1、BgrNdx2、...BgrNdxN として）をスキャンし、dbkey リストをメモリー・バッファまたは一時テーブルに保存します。前の dbkey リストにない dbkey は破棄されるため、各 dbkey リストは通常前のリストより小さくなります。結果として、現在および前回完了した dbkey のリストのみが、常に維持されます。

静的オプティマイザは、2 つ以上のバックグラウンド・インデックスを選択する場合、最初のインデックス・シーケンスは、各インデックスの制限を満足する大まかに見積った dbkey の数をベースにしています。しかし、制限に含まれる変数の値を変えることができるので、実際の dbkey の数は、新しいリーフ・ノードのそれぞれの起動時に変更される可能性があります。

各リーフ起動に対してより適した dbkey の数を見積るため、動的最適化は各リーフ起動時に各バックグラウンド・インデックスに対して迅速に見積りを行う手続きをします。この見積りの目的は、バックグラウンド・インデックスのシーケンスを昇順の選択性順序に変更するためです。この順序は、範囲定義変数を変更できるので、1 つのリーフ起動から別のものを起動するときに変更されることがあります。新しい変数値をベースにした再シーケンスによって、各リーフ起動によって範囲のサイズが大きくなったり小さくなったりしても、最適なインデックスの利用が保証されます。見積り手続きに極端に時間がかかるような場合、動的見積りは行われません。たとえば、次のような場合です。

- 論理 OR を含む範囲リストが、5.7.1 項（5-27）で説明したように制限に含まれている。
- インデックスは 1 つ以上の記憶領域に保存されている。

表 5-1（5-33）には、バックグラウンド・インデックス・スキャンが終了する条件を示しています。このテーブルは、各条件を示す実行トレース出力（RDMS$DEBUG_FLAGS または RDB_DEBUG_FLAGS が E と定義されている場合）も示しています。

表 5-1 バックグラウンド・インデックス・スキャンを終了する条件

条件	E フラグ出力
インデックス範囲全体をスキャンして完了します。	EofData
読み込まれた dbkey が多すぎます（しきい値の限界に達した）。バックグラウンド・プロセスは、他の利用可能なインデックスをスキャンします。	ThreLim
I/O 操作が多すぎます（フェッチの限界）。バックグラウンド・インデックスは次のインデックスを試します。	FtchLim
「Close Leaf」などの明示的なコマンドが発行された。	Termin*

注意： 表 C-3（C-21）には、RDMS$DEBUG_FLAGS および RDB_DEBUG_FLAGS E フラグに対するすべての出力表記法のリストおよび定義があります。

バックグラウンド・プロセスは、呼出し側に列を渡しません（このため、バックグラウンドという用語が使用されています）。バックグラウンド・プロセスが停止したとき、動的リーフレベルの最適化は、通常最終（Fin）ステージに切り替わります。Fin ステージは、次をベースにしてデータ列を渡します。

- メモリー・バッファに収集された dbkey のリスト。これは、ストラテジ出力で「Fin Buf」として記録されます。
- 一時テーブルに収集された dbkey のリスト。これは、「Fin TTbl」として記録されます。

Fin ステージは、別のステージに切り替わりません。そして、明示的なコマンド「Close Leaf (- `CUT)」以外によって割り込まれることはありません。

フォアグラウンド・プロセス

動的リーフレベル最適化の 2 番目のコンポーネントは、フォアグラウンド・プロセスです。これは、フォアグラウンド・インデックスをスキャンするか、バックグラウンド・プロセスから dbkey を取り込みます。フォアグラウンド・プロセスは、バックグラウンド・プロセスと並行して実行します。フォアグラウンド・プロセスは、インデックスから dbkey を読み出

すか、バックグラウンド・プロセスから dbkey を受け取った後すぐに列のフェッチと送出を行います。

5.7.2.1 項（5-34）から 5.7.2.4 項（5-37）は、動的最適化により、データ列の検索に使用される 4 つのリーフ・タイプを説明しています。

5.7.2.1 バックグラウンドのみの検索

動的最適化の最も重要なコンポーネントは、バックグラウンド・プロセスです。このプロセスは、1 つ以上のインデックスをスキャンし、dbkey のソート・リストを昇順で検索の最終（Fin）段階に送出します。Fin ステージは、その後データ列を送出します。バックグラウンドのみのリーフ・ストラテジ（BgrOnly）は、検索の総時間に対するクエリーを最適化します。つまり、BgrOnly は、送出前にクエリーを満足するすべての列を検索します。列をすぐに送出するフォアグラウンド・プロセスは、BgrOnly ストラテジの一部としては実行されません。

オプティマイザは、1 つ以上のインデックスが検索に使用できるが、その中にクエリーが要求する列のすべてを含んでいるものはない場合にバックグラウンドのみのリーフ・ストラテジを選択します。

バックグラウンドのみのリーフ・ストラテジは、次に示す一般的なステップを実行します。

1. 見積った選択性により優先された有用なインデックスをすべてスキャンします。最も短いインデックスが最初にスキャンされます。
2. dbkey リストに dbkey を保存します。
3. dbkey リストの長さがすでに作成された dbkey リストの長さを超えたら、インデックスのスキャンを中止します。収益のないインデックスを中止する決定は、増大する dbkey リストの実際の長さをベースにしており、エラーが発生しやすい選択性見積りによるものではありません。さらに、オプティマイザは各インデックス・スキャンに対する実際のフェッチ（物理的 I/O）をカウントし、現時点で利用できる最高の検索方法に対するコストの比率についての実験を制限します。
4. スキャンを行っている間、事前に作成された dbkey リストに含まれていない dbkey は取り除きます。
5. 最も短く作成された dbkey リストから dbkey を使用して、Fin ステージでの列のフェッチを可能にします。
6. 同じデータ・ページを複数回読み出さないように、最も短くできた dbkey リストを Fin ステージに渡す前にソートします。

バックグラウンド・プロセスは、利用可能なインデックスの中で最高のサブセットの使用を保証することで、総検索時間を最適化します。クエリーの制限事項が 1 つの列をカバーしているか、テーブル列の 90 パーセントをカバーしているか、その制限に対して 1 つ以上のインデックスが存在するかどうかなどについて心配する必要はありません。しかし、列の更新

環境では、セグメントの少ないインデックスを複数定義することは、複数のセグメントを持つインデックスを 1 つ定義するよりも効率的です。

Oracle Rdb を使用すると、クエリー内で総検索時間ストラテジを指定できます。詳細は、5.8.5 項（5-41）を参照してください。

5.7.2.2 高速な最初の検索

BgrOnly リーフの検索ストラテジの総時間と対比して、クエリーまたはアプリケーションによっては、クエリーが返す最初の列または最初の数列のみを必要とします。たとえば、EXISTS 述語は 1 つのデータ列の存在のみを確認します。インタラクティブ・ユーザーは、一般的にクエリーをキャンセルする前に、クエリーの結果生じた多くの画面ではなく最初の画面または最初の数画面を見ます。この場合、オプティマイザは、高速な最初の（FFirst）検索リーフ・ストラテジを使用します。

高速な最初の検索は、次の条件が真の場合に使用します。

- 適切なインデックスが少なくとも 1 つ存在する
- インデックスのみの検索が行えるインデックスが存在しない
- ソートは不要

バックグラウンド・プロセスは直接列を生成できないので、オプティマイザは、FFirst 検索用のバックグラウンド・プロセスをフォアグラウンド・プロセスと並行して実行します。フォアグラウンド・プロセスは、即座に列を作成できます。

FFirst リーフ・ストラテジは、次の一般的な各ステップを実行します。

1. バックグラウンド・プロセスは最適なインデックスをオープンし、スキャンを開始します。バックグラウンド・プロセスは、インデックス・スキャンからフォアグラウンド・プロセスに対して各 dbkey を渡します。
2. フォアグラウンド・プロセスは、すぐに列を作成し始め、作成した列の dbkey をフォアグラウンド・バッファに保存します。レコード検索に対して高速な最初の検索ストラテジが選択された場合に、レコードが選択されなかったり、少数のレコードしか選択されないことがあります。この場合、リーフ・ノードのフォアグラウンド・プロセスは多くのレコードをフェッチして送出しようとしますが、データのすべて（またはほとんど）は選択基準をベースにして却下されます。
3. バックグラウンドおよびフォアグラウンド・アクティビティが行われている間に明示的な Close Leaf コマンドをリーフが受け取る場合、このリーフを呼び出したユーザー（またはユーザーのプログラム）またはクエリーの一部が、すでにフォアグラウンド・プロセスにより送出された少数の列で十分であり、これ以上の列は不要であることを示しています。このように検索が早期に終了すると正確に予測される場合、高速な最初の検索ストラテジはフォアグラウンド・プロセスを使用します。早期に終了すると、バックグラウンド・プロセスとフォアグラウンド・プロセスの両方が停止し、フォアグラウンド・バッファを含むリーフ・コントロール・リソースが解放されます。

4. フォアグラウンド・フェッチのコストが見積られたバックグラウンド・コストの半分を超えたときに、フォアグラウンド・フェッチが終了し、バックグラウンドのみ（BgrOnly）のストラテジに切り替わると、フォアグラウンド・プロセスとバックグラウンド・プロセスの間で競争が起こります。この機能は、フォアグラウンド・プロセスによる成功のチャンスが途方もなく低くなるとすぐに、不必要なフォアグラウンド・コストを削減し、推奨する効率的なストラテジを選択します。ユーザーが選択した高速な最初の最適化が正しくなかったか、以前に行われた未知のデータ配分が総時間の最適化を高速な最初の最適化よりも効率的に行った場合、この競争メカニズムはよりよいパフォーマンスを提供します。
   オプティマイザは、フォアグラウンド・プロセスを終了するときに、フォアグラウンド・バッファを廃棄しません。同じ列を 2 度送出することはできないので、どの列が送出されたかを引き続き知っておく必要があります。バックグラウンドは、列を取得して、その dbkey を Fin ステージに渡し続けます。Fin ステージは、フォアグラウンド・バッファを通して dbkey を抽出し、残りの列を送出します。

5. 明示的な Close Leaf コマンドの前にバックグラウンド・プロセスが終了したら、フォアグラウンドは終了します。バックグラウンド・プロセスは、Fin ステージに切り替わり、列を送出します。

Oracle Rdb を使用すると、クエリー内で高速な最初の検索ストラテジを指定できます。詳細は、5.8.5 項（5-41）を参照してください。

5.7.2.3 インデックスのみの検索

クエリーを満足するために必要なすべての列を含むインデックスがあれば、オプティマイザはインデックスのみ（NdxOnly）のリーフ検索ストラテジを使用します。少なくとも 1 つ、別のインデックスが利用できることが必要です。さもなければ、オプティマイザは従来からのインデックスのみの検索を使用します。

インデックスのみのリーフ・ストラテジは、次の一般的なステップを実行します。

1. オプティマイザは、バックグラウンドおよびフォアグラウンド・プロセスの両方を開始します。フォアグラウンド・プロセスは、クエリーを完了するために必要なすべての列を含むインデックスをオープンします。バックグラウンド・プロセスは、残りの利用可能なインデックスから最も dbkey が少なくてすむと見積ったインデックスをオープンします。オープンすると、両方のプロセスは、物理 I/O 操作で測定されるコストに基づいた一定の速度で処理を進めていきます。

2. フォアグラウンド・プロセスは、すぐに列の作成を開始します。列が送出されたら、オプティマイザは列の dbkey をバッファに記録します。バッファがオーバーフローしたら、バックグラウンド・プロセスは終了し、完了するまでインデックスのみのスキャンを続けます。バックグラウンドの dbkey リストが利用可能になったとき、バッファがオーバーフローしていなければ、最後の列のフェッチに使用されます（NdxOnly スキャンはこのポイントで終了します）。

バックグラウンド・プロセスで何が発生したかを考えてみましょう。バックグラウンドは、最高のインデックスをスキャンしながらゆっくり dbkey リストを構築します。一番の候補を先に行い、同時に 2 つ以上のインデックスをオープンしないようにします。フォアグラウンド・バッファがオーバーフローする前にバックグラウンド・プロセスが dbkey リストの作成を完了すれば、フォアグラウンド・プロセスは終了します。バックグラウンド・プロセスは列を送出できないので、オプティマイザは、Fin ステージに移動します。同じデータを 2 度出力しないようにするため、バックグラウンドの dbkey リストからの dbkey は、フォアグラウンド・バッファに記録されているすでに出力されたリストに対して確認されます。フォアグラウンド・バッファに存在する列と一致しない列は、呼出し元に送出されます。

5.7.2.4 ソート順序検索

オプティマイザは、次の条件が存在する場合、ソート順序リーフ検索ストラテジを使用します。

- クエリーがあるソート順序を指定または暗示する場合、たとえば、集計、統計または一致ストラテジで特定のソート順序の必要性がある場合です。
- インデックスのみの検索が不可能である場合。
- 正しい順序のインデックスが存在する場合。
- 少なくとも 1 つ、別のインデックスが存在し、dbkey の抽出に使用できる場合。

ソート順序リーフ・ストラテジは、次の一般的なステップを実行します。

1. オプティマイザは、バックグラウンドおよびフォアグラウンド・プロセスの両方を開始します。フォアグラウンド・プロセスは、ソート・インデックスをオープンし、すぐに列の送出を開始します。送出した列の dbkey は、バッファ内に保存されません。バックグラウンド・プロセスは、フォアグラウンド・プロセスと並行して他の有用なインデックスのスキャンを開始します。バックグラウンドは、フォアグラウンド・プロセスと競争しませんが、そのかわり、抽出する目的で、dbkey リストまたはビットマップを構築します。
2. バックグラウンド・プロセスが終了したら、すべてのリソースを解放し、最終的な dbkey リストか最終的なビットマップ・フィルタのみを保持します。
3. フォアグラウンド・プロセスが送出し続ける列は、バックグラウンド dbkey リストまたはビットマップに対してチェック（フィルタ）されます。dbkey が一致すれば、その列はデータ・ページから検索されます。一致しなければ、列はフェッチされず、I/O を節約します。

5.8 クエリー・オプティマイザでの作業

この項では、オプティマイザの可能性を最大限にする方法を説明します。

5.8.1 ビューとクエリーの最適化

ビューが十分に有効に定義されている場合には、ビューを使用して、オプティマイザが最高の検索ストラテジを開発するのを支援できます。

ビューは、最適化作業を容易にする機会を提供するので、検索ストラテジの決定に要する総時間を削減できます。3 テーブルのビューと 4 つめのテーブルをクエリーで結合した場合、オプティマイザは、ビューに対する検索ストラテジを導出し、次にクエリーに対する検索ストラテジを導出します（これが、ビュー結合テーブルです）。ビューは、ビューのデータ・ストリームの結果として見積られたカーディナリティを持つ、1 つのテーブルとして取り扱われます。言い換えると、ビュー検索ストラテジは、残りのクエリーとは別であると判断されます。効果的なビュー検索ストラテジを生成するため、ビュー列をベースにしている述語だけでなく、ビューの外部にある述語も使用します。同じように、クエリー用の検索ストラテジの生成では、ビューのカーディナリティ見積りが使用されます。ビューはクエリーとは別に最適化されますが、該当する場合にはグローバルな情報が使用されます。

次の例を考えてみましょう。TAB_A、TAB_B、TAB_C および TAB_D の 4 つのテーブルを結合する必要があります。TAB_A、TAB_B および TAB_C がビューで結合されている場合、オプティマイザが考える必要がある可能なソリューションの数（インデックスも等価カーディナリティもないと仮定すると）は、ビューが使用されておらず、4 つのテーブルがすべて等価なカーディナリティを持ち、インデックスが結合されていない場合の 4!（または（4 × 3 × 2 × 1 = 24））に対して、3! プラス 2!（または（3 × 2 × 1 + 2 × 1 = 8））です。しかし、ソリューション・プルーニングが発生するので、各ソリューションの検討の程度が異なると考えられます。ビューが使用されると、可能性があるソリューションの数は減少するので、クエリーを最適化する時間の総計も減少します。

他の例を考えてみましょう。TAB_A、TAB_B、TAB_C および TAB_D の 4 つのテーブルがあります。TAB_A と TAB_B は VIEW_1 に結合され、TAB_C と TAB_D は VIEW_2 に結合されており、この 2 つのビューの間には結合がない場合、オプティマイザは、VIEW_1（最初の結合）に対する検索ストラテジ、VIEW_2（2 番目の結合）に対する検索ストラテジおよび VIEW_1 と VIEW_2 の結果のクロス生成に対する検索ストラテジを出します。しかし、4 つのテーブルのすべてがクエリーによって使用されている場合、オプティマイザは 2 つのテーブルを結合する検索ストラテジを生成します。続いて 3 つめのテーブルとのクロス積のストラテジ、最後に 4 つめのテーブルとの結合ストラテジを生成します。2 番目のケースでは、クロス積は実行される最後の操作ではないので、そのような検索ストラテジのパフォーマンスは、ビューを使用してクエリー用に生成された検索ストラテジよりも悪くなることもあります。

5.8.2 連結式とクエリーの最適化

連結式の間の比較を含むクエリーは、場合によってはインデックスが使用できるような等価な式に分解されます。次に 2 つの可能性があるケースを示します。

```
SELECT * from table where a || b = c || d;              (case A)
SELECT * from table where a || b > 'Some string';       (case B)
```

これらの各ケースで、a が固定幅のテキスト列でインデックスが a で定義されている場合、そのインデックスは利用可能なはずです。A の場合、列 a の幅は列 c の幅と同じであることが必要です。

1 つ以上のインデックスにインデックスの先頭のセグメントとして a または b が含まれていれば、高速インデックス参照を使用できます。高速インデックス参照は、フル・インデックス・スキャンのパフォーマンスよりも高速です。

次の制限は、連結式の最適化に適用されます。

- a と :aVar の幅は同じであることが必要です。（この制限はケース A に適用されます。）
- 利用可能なインデックスが列 a 上に定義されていることが必要です。（この制限はケース A に適用されます。）
- 列 a は、固定幅のテキスト列であることが必要です。（この制限はケース A および B に適用されます。）

5.8.3 最初のキー・セグメントを直接ベースにしていないクエリー

避けるべき（または少なくとも重要性がわかっている）状況は、次の特徴を持つクエリーです。

- 直接的または間接的な OR 論理の使用
- セグメント・キーを持つインデックスの使用
- 最初のキー・セグメントに直接依存

AND 論理を使用するクエリーは、AND を使用したほうがより小さい dbkey のサブセットを生成するので、OR 論理を使用するクエリーよりも常によく動作します。OR 論理は、OR リストを連結し、結果として生じたリストは 2 つのリストの合計に等しくなります。表 5-2 (5-39) は、AND と OR 論理の結果を示します。

表 5-2 AND と OR の論理の比較

条件	クエリーの数	dbkey 設定サイズ
AND	1	小さい
OR	2	大きい、連結

セグメント・キーに対して、一意に行を識別するために 2 つ以上の列を使用すると、セグメントをクエリー内でどのように表すか注意する必要があります。たとえば、次の 3 つの部分からなるキー名 EMP_JOB_DEPT を考えてください。

- EMPLOYEE_ID 列は、最初のキー・セグメントを形成します。
- JOB_CODE 列は、2 番目のキー・セグメントを形成します。

- DEPARTMENT_CODE 列は、3 番目のキー・セグメントを形成します。

クエリーはこのキーの最初のセグメントをベースにして、OR 論理を使用して 2 番目と 3 番目のセグメントを含めます。最高のアプローチは、有害な OR 条件を避けることです。キーの最初のセグメントと、OR 条件を持つ残りの 2 つのセグメントからなる AND 条件を使用したクエリーを形成すると、問題が発生する可能性があります。これを数学的に表現すると、次のようになります。

```
Where, S1=EMPLOYEE_ID, S2=JOB_CODE, and S3=DEPARTMENT_CODE
S1 = X AND ((S2 = Y AND S3 = Z) OR (S2 =Y1 AND S3 = Z1))
```

この方法で、クエリーを表すと、外側のカッコ内の文、(S2 = Y AND S3 = Z) OR (S2 =Y1 AND S3 = Z1) はセグメント 1 を直接ベースにしていません。そのかわり、セグメント 2 と 3 の間に OR 条件があり、この結果がセグメント 1 をベースにしてインデックス検索されます。セグメント 2 と 3 がセグメント 1 を直接ベースにしていることを確実にするには、次の方法で、クエリーで表現することをお薦めします。

```
(S1 = X AND S2 = Y AND S3 = Z) OR (S1 =X AND S2= Y1 AND S3 = Z1)
```

要約すると、最初のキー・セグメントを直接ベースにしていないクエリーの使用は避ける方が優れています。

5.8.4 インデックスの配置

テーブルの CREATE STORAGE MAP 文に PLACEMENT VIA 句が指定されている場合のみ、またはもう 1 つのソート・インデックスが PLACEMENT VIA インデックスの先頭セグメントに等しい 1 つ以上の先頭セグメントを持つ場合に、動的最適化は、データ行がソート・インデックスによってクラスタ化されていると仮定します。その他の非 PLACEMENT VIA インデックスのすべては、クラスタ化効果を持たないと見なされます。つまり、ランダムな物理行のフェッチはそのようなインデックス・スキャンから得られるので、少なくとも一度の I/O 操作によって 1 行がフェッチされます。PLACEMENT VIA インデックスを定義した後それを削除した場合、または非 PLACEMENT VIA インデックスが PLACEMENT VIA インデックスに対して強い統計上の相互関係を持っている場合、またはテーブルがあるインデックス順序と同じシーケンスでロードされた場合、このような暗黙的なクラスタ化効果はオプティマイザに対して隠蔽されます。

よりよいパフォーマンスを保証するには、クラスタ化順序を明示的に指定する必要があります。しかし、暗黙的なクラスタ化であっても、同じインデックスに対して 2 つ以上のインデックス・スキャンが実行されると、動的なオプティマイザは、クラスタ化の要因を検出し測定します。これは、コンパイルされた同じクエリーを複数回実行するか、1 つのクエリーの実行で複数のインデックス検索を実行する場合、オプティマイザは任意のクラスタ化を適合させ、最高のパフォーマンス率に近づけることを意味しています。

5.8.5 優先最適化モードの指定

高速な最初の行検索と総時間検索のいずれかを指定した場合、アプリケーションと 4GL ツールは、クエリー・オプティマイザによって選択された検索ストラテジに影響を与えます。

- 高速な最初の行検索では、データは可能な限り迅速に返されます。このストラテジは、クエリーを満足する 1 つまたは複数の行の存在を確認する必要があるか、または検出する必要があるアプリケーションにとって役立ちます。たとえば、対話型アプリケーションでは、データのいくつかを画面に表示したら、すべての行のセットを読み込まなくても、ユーザーはクエリーを中止することができます。高速な最初の検索ストラテジの詳細は、5.7.2.2 項（5-35）を参照してください。
- 総時間検索の場合、オプティマイザはクエリー全体を満足するための、最高のストラテジを決定します。このストラテジは、総検索時間の最小化に興味があり、最初の行のデータが戻ってくるまで待てるバッチ・ジョブなどのアプリケーションに役立ちます。総検索時間は、常にバックグラウンドのみの検索ストラテジで使用されます。バックグラウンドのみの検索ストラテジの詳細は、5.7.2.1 項（5-34）を参照してください。

優先オプションの存在にかかわらず、Oracle Rdb は、確たる理由を検出したら、特定のオプションを無視できます。たとえば、EXISTS 述語は、クエリーに対して総時間オプションが指定されているかどうかにかかわらず、高速な最初の最適化を要求します。GROUP BY 句を持たない集計式（AVG、COUNT、MAX、MIN および SUM）は、クエリーに対して高速な最初の検索オプションが指定されているかどうかにかかわらず、総時間最適化を要求します。

SQL を使用すると、次を使用してオプティマイザ・プリファレンスを指定できます。

- SELECT 式
- 1 行形式の SELECT 文
- プリコンパイラ・コマンドライン上の SQLOPTIONS 修飾子
- モジュール言語コマンドライン上の OPTIMIZATION_LEVEL 修飾子
- SET OPTIMIZATION LEVEL 文

SELECT 式と 1 行形式の SELECT 文は、個々の文に対してオプティマイザ・プリファレンスを指定するために使用します。プリコンパイラ・コマンドライン上の SQLOPTIONS=OPTIMIZATION_LEVEL 修飾子は、プリコンパイラ・プログラム内の大部分の文に対してオプティマイザ・プリファレンスを指定するために使用します。モジュール言語コマンドライン上の OPTIMIZATION_LEVEL 修飾子は、モジュール言語コマンドライン内の大部分の文に対してオプティマイザ・プリファレンスを指定するために使用します。SET OPTIMIZATION LEVEL 文は、動的 SQL または対話型 SQL 内の大部分の文に対してオプティマイザ・プリファレンスを指定するために使用します。

この項ではこれ以降、SQL を使用したオプティマイザ・プリファレンスの指定方法の詳細情報を説明します。構文の詳細は、『Oracle Rdb7 SQL Reference Manual』を参照してください。

- SELECT 式
  Oracle Rdb が SELECT 式に対して使用する最適化モードを指定するために、OPTIMIZE FOR { DEFAULT | FAST FIRST | TOTAL TIME } 句を SELECT 式で指定できます。SELECT 式は、DECLARE CURSOR 文、INSER 文および対話型 SELECT 文とともに使用できることに注意してください。
  次の対話型 SQL の例は、カーソルに対してよりよい最適化モードを設定する 2 つの DECLARE CURSOR 文を示しています。最初の DECLARE CURSOR 文は、高速な最初の検索に対する最適化モードを設定します。2 番目の DECLARE CURSOR 文は、総時間検索に対する最適化モードを設定します。

```
SQL> DECLARE CEMP TABLE CURSOR
  1>  FOR
  2>    SELECT    *
  3>      FROM   EMPLOYEES
  4>      WHERE  EMPLOYEE_ID > '01100'
  5>  OPTIMIZE FOR FAST FIRST;
SQL>
SQL> DECLARE TEMP TABLE CURSOR
  1>  FOR
  2>    SELECT    LAST_NAME, FIRST_NAME
  3>      FROM   EMPLOYEES
  4>  OPTIMIZE FOR TOTAL TIME;
```

  次の対話型 SQL の例では、SELECT 文に対する最適化ストラテジを選択する方法を示しています。

```
SQL> SELECT * FROM EMPLOYEES
  1> OPTIMIZE FOR FAST FIRST;
SQL>
SQL> SELECT * FROM EMPLOYEES
  1> OPTIMIZE FOR TOTAL TIME;
```

- 1 行形式の SELECT 文
  OPTIMIZE FOR { DEFAULT | FAST FIRST | TOTAL TIME } 句は、Oracle Rdb が 1 行形式の SELECT 文に対して使用する最適化モードを、1 行形式の SELECT 文で指定できます。1 行形式の SELECT 文は、SQL モジュール言語と SQL プリコンパイラ・プログラムでのみ有効であることに注意してください。

- SQLOPTIONS=(OPTIMIZATION_LEVEL=DEFAULT | FAST_FIRST | TOTAL_TIME)
  SQL プリコンパイラ・コマンドライン用の SQLOPTIONS=(OPTIMIZATION_LEVEL=DEFAULT | FAST_FIRST | TOTAL_TIME) 修飾子は、OPTIMIZE FOR 句で特定のオプティマイザ・ストラテジを指定しない SQL プリコンパイラ・プログラムに組み込まれた文を処理するためにのデフォルトのオプティマイザ・ストラテジを指定します。
  SQLOPTIONS=(OPTIMIZATION_LEVEL=DEFAULT | FAST_FIRST | TOTAL_TIME)

修飾子を使用して、プリコンパイラ・プログラム内のほとんどの文に対して適切なオプティマイザ・ストラテジを選択できます。オプティマイザ・ストラテジを指定するために OPTIMIZE FOR 構文を使用するプリコンパイラ・プログラム内の個々の文は、OPTIMIZE FOR 構文で指定したオプティマイザ・ストラテジで処理されます。

- OPTIMIZATION_LEVEL=(DEFAULT | FAST_FIRST | TOTAL_TIME)
  SQL モジュール言語コマンドライン用の OPTIMIZATION_LEVEL=(DEFAULT | FAST_FIRST | TOTAL_TIME) 修飾子は、OPTIMIZE FOR 句で特定のオプティマイザ・ストラテジを指定しない SQL モジュール言語プログラムに組み込まれたこれらの文を処理するために使用されるデフォルトのオプティマイザ・ストラテジを指定します。OPTIMIZATION_LEVEL=(DEFAULT | FAST_FIRST | TOTAL_TIME) 修飾子を使用して、モジュール言語プログラム内のほとんどの文に対して適切なオプティマイザ・ストラテジを選択します。オプティマイザ・ストラテジを指定するために OPTIMIZE FOR 構文を使用しているモジュール言語プログラム内の個々の文は、OPTIMIZE FOR 構文で指定したオプティマイザ・ストラテジで処理されます。

- SET OPTIMIZATION LEVEL { 'DEFAULT' | 'FAST FIRST' | 'TOTAL TIME' }
  動的 SQL の文に対するデフォルトの最適化モードは、文が準備されているモジュールに対して指定されたモードです。SET OPTIMIZATION LEVEL 文は、モジュールに対して設定されたデフォルトの最適化モードを無効にするために使用します。
  対話型 SQL では、デフォルトの最適化モードは DEFAULT です。SQL SET OPTIMIZATION 文は、デフォルトの最適化モードを無効にするため、対話型 SQL で使用できます。
  次に示す対話型 SQL の例は、1 つ以上のクエリーに対して高速な最初の検索ストラテジまたは総時間検索ストラテジを要求するために SET OPTIMIZATION LEVEL 文をどのように使用できるかを示します。
  すべての結果レコードが送出される前に、対話型セッションの大部分のクエリーがクローズされない場合には、次の文を使用します。

```
SQL> SET OPTIMIZATION LEVEL 'TOTAL TIME';
```

  最初の数レコードを検査した後（すべての結果レコードが送出される前）で対話型セッションの大部分のクエリーが終了する場合は、次の文を使用します。

```
SQL> SET OPTIMIZATION LEVEL 'FAST FIRST';
```

  次の文は、デフォルトの動作に戻す最適化を設定します（SET OPTIMIZATION LEVEL ストラテジが指定されていない場合）。デフォルトの動作は、高速な最初の検索ストラテジを先に行ってみて、高速な最初の検索ストラテジよりも総時間検索ストラテジのほうが高速にレコードを検索するようであれば、その後で総時間検索ストラテジを選択します。

```
SQL> SET OPTIMIZATION LEVEL 'DEFAULT';
```

  対話型のセッションの異なる部分で異なる最適化レベルが必要な場合、SQL SET OPTIMIZATION LEVEL 文を使用して各部分に対して望むレベルを設定します。
  例 5-3（5-44）では、SET OPTIMIZATION LEVEL 文の使用を示しています。この例は、

RDMS$DEBUG_FLAGS 論理名または RDB_DEBUG_FLAGS 構成パラメータを S と定義すると仮定しています。

例 5-3 SQL SET OPTIMIZATION LEVEL 文の使用

```
$ SQL
SQL> ATTACH 'FILENAME personnel';
SQL> --
SQL> -- No optimization level has been selected.  The optimizer
SQL> -- selects the fast first (FFirst) retrieval strategy to
SQL> -- retrieve the rows from the EMPLOYEES table in the
SQL> -- following query:
SQL> SELECT EMPLOYEE_ID, LAST_NAME
  1>  FROM EMPLOYEES
  2>  WHERE EMPLOYEE_ID IN ('00167', '00168');
Leaf#01 FFirst RDB$RELATIONS Card=19
  BgrNdx1 RDB$REL_REL_NAME_NDX [1:1] Fan=8
Sort
Cross block of 2 entries
  Cross block entry 1
    Leaf#01 BgrOnly RDB$RELATION_FIELDS Card=71
      BgrNdx1 RDB$RFR_REL_NAME_FLD_ID_NDX [1:1] Fan=8
  Cross block entry 2
    Get     Retrieval by index of relation RDB$FIELDS
      Index name  RDB$FIELDS_NAME_NDX [1:1]  Direct lookup
Leaf#01 FFirst EMPLOYEES Card=100
  BgrNdx1 EMP_EMPLOYEE_ID [1:1...]2 Fan=17
 EMPLOYEE_ID   LAST_NAME
 00167         Kilpatrick
 00168         Nash
2 rows selected
SQL> --
SQL> -- Use the SET OPTIMIZATION LEVEL statement to specify that you want
SQL> -- the total time (BgrOnly) retrieval strategy to be used.  Note that
SQL> -- when the previous query is executed again, the total time (BgrOnly)
SQL> -- retrieval strategy is selected, instead of fast first:
SQL> SET OPTIMIZATION LEVEL 'TOTAL TIME';
SQL> SELECT EMPLOYEE_ID, LAST_NAME
  1>  FROM EMPLOYEES
  2>  WHERE EMPLOYEE_ID IN ('00167', '00168');
Leaf#01 BgrOnly EMPLOYEES Card=100
  BgrNdx1 EMP_EMPLOYEE_ID [1:1...]2 Fan=17
 EMPLOYEE_ID   LAST_NAME
 00167         Kilpatrick
 00168         Nash
2 rows selected
SQL> --
SQL> -- When you specify the SET OPTIMIZATION LEVEL 'DEFAULT' statement,
```

```
SQL> -- either the fast first or total time strategy will be selected.
SQL> -- The fast first strategy will be tried first, then total time
SQL> -- will be selected if it will retrieve the rows faster than the
SQL> -- fast first strategy.
SQL> SET OPTIMIZATION LEVEL 'DEFAULT';
SQL> --
SQL> -- Because the fast first strategy is faster than the total
SQL> -- time strategy for this query, the fast first strategy
SQL> -- is used to retrieve the rows:
SQL> SELECT EMPLOYEE_ID, LAST_NAME
  1>  FROM EMPLOYEES
  2>  WHERE EMPLOYEE_ID IN ('00167', '00168');
Leaf#01 FFirst EMPLOYEES Card=100
  BgrNdx1 EMP_EMPLOYEE_ID [1:1...]2 Fan=17
 EMPLOYEE_ID   LAST_NAME
 00167         Kilpatrick
 00168         Nash
2 rows selected
SQL>
```

初期化手続き（各セッションの最初で自動的に実行される sqlini.sql 手続き）を使用する場合、最も一般的に使用する最適化レベルを初期化手続きに設定できます。

5.8.6 クエリー・ガバナーの使用

データベース環境によっては、対話型ユーザーまたはアプリケーション・ユーザーは、複数のテーブル結合を必要としたり、テーブルのすべての行を戻すような一般的なクエリーを入力することで、過大なシステム・リソースを消費することがあります。デフォルトでは、Oracle Rdb は各クエリーが完了するまで実行します。アプリケーションや環境内で、冗長なクエリーに対してクエリー出力の制限を設定することで、オーバーロードを防ぐことができます。出力を制限するメカニズムは、**クエリー・ガバナー**と呼ばれます。クエリー・ガバナーを使用すると、次に示す 1 つ以上の制限を設定できます。

- クエリーが戻す行の数の制限を決めることで、出力を制限できます。オプティマイザは、クエリーが戻す各行をカウントして、行の制限に達したら、実行を停止します。行の制限は、結合のための中間行や集計用のコンポーネント行など、アプリケーションが読み込む行の数とは独立していることに注意してください。しかし、行の制限値は、クエリーの構文解析に必要なメタデータをフェッチするシステム・テーブル（たとえば、SQL による）のクエリーを含むすべてのデータベース・クエリーに適応されるので、極端に小さな値の設定は避けてください。行の制限値が必要なメタデータに対するアクセスを阻止する場合、実行しようとしたクエリーは失敗します。
- オプティマイザがクエリーのコンパイルに消費する時間の総計を制限することで、出力を制限できます。制限時間の値は、消費した秒数を示す整数です。

注意： クエリー・コンパイルの制限時間を指定すると、環境によってはアプリケーション障害が発生します。システムの負荷が軽い閑散時は実行に成功したアプリケーションでも、ピーク時に実行すると、タイムアウトすることがあります。

- 実行するクエリーを最適化するために、CPU 時間の総計を制限できます。CPU 制限時間の値は、CPU 秒数を指定する整数です。デフォルトでは、クエリーのコンパイルに使用する CPU 時間には制限がありません。

3 つのすべてのケースで、制限を超えると、ユーザーはエラー・メッセージを受け取ります。たとえば、ユーザーが行の制限を越えた場合、Oracle Rdb は次のメッセージを表示します。

```
%RDB-E-EXQUOTA, Oracle Rdb runtime quota exceeded
-RDMS-E-MAXRECLIM, query governor maximum limit of records has been reached
```

次のいずれかの方法を使用して、行制限値と制限時間値を設定できます。

- 対話型 SQL の場合、SET QUERY LIMIT 文を使用します。

```
SQL> SET QUERY LIMIT TIME h
SQL> SET QUERY LIMIT ROWS i
SQL> SET QUERY LIMIT CPU TIME j
```

各制限に対して設定された値を表示するには、SHOW QUERY LIMIT 文を使用できます。

```
SQL> SHOW QUERY LIMIT
  QUERY LIMIT TIME limit is h seconds
  QUERY LIMIT ROWS limit is i rows
  QUERY LIMIT CPU TIME limit is j seconds
```

- SQL プリコンパイラの場合、SQLOPTIONS 修飾子に対してクエリー制限パラメータを使用します。

```
/SQLOPTIONS = (QUERY_TIME_LIMIT = h, QUERY_MAX_ROWS = i,
                        QUERY_CPU_TIME_LIMIT = j)
```

- SQL モジュール言語プロセッサの場合、次の修飾子を使用します。

```
/QUERY_TIME_LIMIT = h  /QUERY_MAX_ROWS = i  /QUERY_CPU_TIME_LIMIT = j
```

- 動的 SQL の場合、オプションはコンパイル修飾子から継承します。
- 論理名 RDMS$BIND_QG_TIMEOUT、RDMS$BIND_QG_REC_LIMIT および RDMS$BIND_QG_CPU_TIMEOUT または構成パラメータ RDB_BIND_QG_TIMEOUT、RDB_BIND_QG_REC_LIMIT および RDB_BIND_QG_CPU_TIMEOUT を使用します。詳細は、A.85 項（A-39）, A.84 項（A-38）および A.83 項（A-38）をそれぞれ参照してください。

OpenVMS VAX OpenVMS Alpha

RDO を使用している場合、クエリーに制限を加える唯一の方法は、これらの論理名を定義することです。◆

クエリー・ガバナー・オプションのすべてで、制限時間、行の制限および CPU 制限時間を設定できます。いずれかの限界値に達すると、出力は停止します。

5.8.7 RDMS$DEBUG_FLAGS および RDB_DEBUG_FLAGS の使用方法

論理名 RDMS$DEBUG_FLAGS または構成パラメータ RDB_DEBUG_FLAGS を使用して、オプティマイザがクエリーを実行する方法を検査できます。クエリー統計とクエリー・ストラテジを分析すると、パフォーマンスを改良できます。この項では、RDMS$DEBUG_FLAGS 論理名または RDB_DEBUG_FLAGS 構成パラメータを使用する場合、探す必要があるものを説明します。完全な説明と出力のサンプルについては、付録 C（C-1）を参照してください。

順次検索

クエリーのパフォーマンスを改良するには、RDMS$DEBUG_FLAGS または RDB_DEBUG_FLAGS ストラテジ・ディスプレイ（S フラグ）の表記法 "Retrieval sequentially" に特別な注意を払ってください。この表記法は、オプティマイザが順にテーブル全体を検索することを示しています。クエリーのベースとなる列または複数の列に対してインデックスを定義することで、パフォーマンスを改善できます。しかし、単にインデックスを定義するだけで、自動的にクエリーの速度が上がったり、また定義したインデックスをオプティマイザが使用するとは考えないでください。場合によっては、行を順次アクセスするほうが速いため、オプティマイザはインデックスを無視することもあります。順次アクセスは、一般的に安定した小さなテーブルで、重複のない一意な列値が少数の場合によりよく動作します。確信が持てなければ、インデックスを定義して、RDMS$DEBUG_FLAGS か RDB_DEBUG_FLAGS を使用してテストしてください。

ソート

オプティマイザがソートを実行していることを示す表記法の存在にも注意してください。送出前に行をソートすると、クエリーの実行時間が低下します。適切なインデックスを定義することで、行のソートを防ぐことができるので、クエリーの実行時間を短縮できます。

クエリーに ORDER BY 句が含まれていなければ、オプティマイザは特定の行の順序を保証しません。オプティマイザは、クエリーを満足するすべての行の検索を単に見つけるだけです。任意の列に対してある順序を指定する必要がある場合、クエリーに ORDER BY 句を常に含める必要があります。

降順のインデックスが存在しなければ、ソートに余分なステップが必要になるので、降順ソートは避けてください。順番は、どの列にインデックスがあるかによって決まります。列に対してインデックスを定義する場合、Oracle Rdb はインデックス内のノードを値の昇順に並べます。したがって、インデックス付き列に表示されたデフォルトのソート順序は、ASCENDING です。

クエリー・コストのチェック

RDMS$DEBUG_FLAGS か RDB_DEBUG_FLAGS 統計表示（O フラグ）を使用すると、クエリーのコストをチェックして、I/O 操作がどれだけ必要かを決定できます。詳細は、C.4 項（C-15）を参照してください。この情報は、多くのテーブルを結合するような非常に複雑なクエリーを開発している場合に役立ちます。ある列に対してインデックスを定義して、選択した式で名前を付けたり、クエリーを分割してより小さく、簡単なクエリーにすることによって、異なる形式のクエリーで実験することができます。そして、S フラグや O フラグを使用してクエリーを実行できるので、クエリーの各書式に対してクエリー・オプティマイザが選択したアクセス・ストラテジをコストで比較できます。最も低い関連コストを表示したクエリーの書式を使用します。最高のソリューションは、常にクエリーをできる限り簡単にすることです。

一般的には、特定のクエリーが問題を発生させる場合にのみ、クエリー実行コストを心配する必要があります。データベースは、1 つのクエリーの実行から次の実行までの間にかなり簡単に変更されるので、オプティマイザは各実行に対して異なるアクセス・ストラテジを選択します。しかし、複雑な選択式を含む特別なケースでは、ホスト言語プログラムに記述する前に、この種の分析によって利点を得ることができます。

5.8.8 クエリー・コスト見積りの使用

オプティマイザが生成したクエリーのコストについての情報にアクセスでき、これらのコストの見積りをアプリケーション内や対話的に使用することができます。戻ってきた見積りは、クエリーが必要な I/O 操作がどれだけ必要か、いくつの行が送出されるかを示しています。これらのコスト値は、見積られたものであり、RDMS$DEBUG_FLAGS または RDB_DEBUG_FLAGS を O と定義することによって、返された値と同じです。

対話型 SQL 文 SET QUERY CONFIRM を入力して、クエリーを実行した場合、SQL はそのクエリーのコストを表示して、このクエリーをキャンセルするかどうかを問い合せます。これは、例 5-4（5-48）に示します。

例 5-4 SET QUERY CONFIRM 文の使用

```
SQL> SET QUERY CONFIRM
SQL> SELECT   * FROM  EMPLOYEES
  1>   WHERE EMPLOYEE_ID > '01100';
Estimate of query cost:  567 I/Os, rows to deliver:  14
Do you wish to cancel this query (No)?
```

コストが過大であれば、[YES] を入力して、[Enter] を押し、クエリーをキャンセルします。クエリーの実行を継続して、クエリーの結果を表示するには、[Enter] を押します。

オプティマイザは、カーソルをオープンしたときにクエリーのコスト情報も SQL Communications Area（SQLCA）に書き込みます。例 5-5（5-49）に、対話型 SQL DECLARE CURSOR 文内で使用された例 5-4（5-48）の選択式を示します。

例 5-5 SQLCA を通したアクセス・コストの見積り

```
SQL> DECLARE CEMP TABLE CURSOR
  1> FOR
  2>     SELECT   *
  3>       FROM  EMPLOYEES
  4>       WHERE EMPLOYEE_ID > '01100';
SQL> OPEN CEMP;
SQL> SHOW SQLCA
SQLCA:
        SQLCAID:        SQLCA           SQLCABC:        128
        SQLCODE:        0
        SQLERRD:        [0]: 0
                        [1]: 0
                        [2]: 35
                        [3]: 29
                        [4]: 0
                        [5]: 0
        SQLWARN0:               SQLWARN1:               SQLWARN2:
        SQLWARN3:               SQLWARN4:               SQLWARN5:
        SQLWARN6:               SQLWARN7:
        SQLSTATE:       00000
SQL>
```

コストの見積りは、SQLERRD 配列に表示されます。

- SQLERRD 要素 [2] には、見積られた行の数が含まれています。
- SQLERRD 要素 [3] には、見積った I/O 操作の回数が含まれています。

オプティマイザが見積りを提供できなければ、"unknown" を示す値「-1」が返されます。

SQL$PRE または SQL$MOD を使用して見積りにアクセスするには、コマンドラインで QUERY_ESTIMATES 修飾子を使用する必要があります。

アプリケーションで SQLCA 情報を使用する方法については、『Oracle Rdb7 SQL プログラミングのガイド』と『Oracle Rdb7 SQL Reference Manual』を参照してください。

注意： 新しい SQLCA 配列要素をプリコンパイルかモジュール言語プログラムで使用するには、再コンパイルする必要があります。アプリケーションから Oracle Rdb4.1 以前のバージョンが動作しているリモート・システム上の SQLERRD 要素 [2] と [3] にある SQLCA 情報にアクセスしようとしても、このクエリーは実行できません。

5.8.9 制約 BLR は実際の実行ストラテジを反映しない

Oracle Rdb がプライマリ・キー制約クエリーまたは外部キー制約クエリーをコンパイルする場合、Oracle Rdb の制約操作コードは、多くの場合クエリー構造を再フォーマットして、オプティマイザが制約を評価するため最も効率的なストラテジを生成できるようにします。制約クエリーの再フォーマットは、Oracle Rdb の内部で行われるので、Oracle Expert for Rdb のように、ユーザーや他のソフトウェア製品からは見ることができません。

制約クエリーの内部の再フォーマットの結果として、制約の評価に使用する実際の実行ストラテジは、内部の再フォーマットがまったく行われなかった場合のものとは大きく異なります。制約バイナリ言語表現（BLR）は、単に主キーまたは外部キー制約クエリーの標準形式を示します。これは、Oracle Rdb による内部の再フォーマットを反映したものではありません。したがって、Oracle Rdb が制約の評価に使用する実行ストラテジについての結論を、制約 BLR から引き出すことはできません。

5.8.10 その他のヒント

次に、オプティマイザのパフォーマンスを最大限に引き出すその他のヒントを示します。

- テーブル・カーディナリティの見積りに依存する動的最適化（RDB$RELATIONS システム・テーブルの RDB$CARDINALITY 列）。この見積りは、正しい値から大きく外れることはありません。カーディナリティ値の修正の詳細は、5.3.1 項（5-4）を参照してください。
- 特定のデータ値の存在を確認する場合は、COUNT よりも ANY または EXISTS を使用。真の条件が存在すると ANY と EXIST は即座に停止しますが、COUNT は Select 式にあるすべての行（または少なくともすべてのインデックス値）を読み込む必要があります。
- 選択性要因が低いため、NOT、CONTAINING または MATCHING の使用は避ける。インデックスを使用するか使用しないは、クエリーの残りの部分に依存します。
- 更新が多い環境ではインデックスを付けすぎない。インデックスを付けすぎないようにする最大の理由は、新しい行の保存、行の削除および既存の行のキー値の変さらに時間がかかりすぎるためです。さらにインデックスの更新によって発生する障害には、インデックス・レベルのロックがあります。
  しかし、読取り専用環境では、インデックスが急増しても差障りがなく、実際に最適化が改善されます。唯一の制限要因は、ディスク領域の使用率、初期テーブルのロードおよび静的オプティマイザ処理が遅くなる可能性があるということです。
- インデックス・セグメントでは、VARCHAR データ・タイプ列を使用しない。オプティマイザは、これらのインデックス・セグメントではインデックスのみの検索を行えません。
- ビューやサブクエリーをむやみに使用しない。使いすぎると、オプティマイザの動作を妨げることになります。ビューもサブクエリーもそれぞれの役割があります。疑問があれば、他のストラテジを試して、データやアプリケーションに適したソリューションを見つけるためにデバッグ・フラグを使用して情報を収集してください。

5.9 クエリーの安定性、制御性およびパフォーマンスをクエリーのアウトラインを使用して確認

Oracle Rdb バージョン 6.0 以前のバージョンでは、オプティマイザがクエリーに対して選択したストラテジを直接制御する方法はありませんでした。これは、安定性と制御性の欠如は、クエリーの最適化プロセスに固有のものであったことを意味しています。

安定性とは、Oracle Rdb のバージョンに渡ってクエリーの予測されるパフォーマンスが維持されていることを意味しています。オプティマイザは、Oracle Rdb の各バージョンで変更されています。オプティマイザに対するこれらの変更のため、クエリーによっては、あるバージョンの Oracle Rdb では受け入れられても、より新しいバージョンの Oracle Rdb では実行状態が悪化することがあります。

制御性は、手動による結合順序、結合方法およびクエリーに対するインデックスの使用を指定する機能を示します。Oracle Rdb バージョン 6.0 より前のバージョンでは、実行できなかったクエリーを書き直すことにより、クエリーの最適化で異なる結合順序、結合方法またはインデックスを使用するように、オプティマイザに影響を与えることができます。しかし、結合順序、結合方法、オプティマイザが使用するインデックスを直接指定することはできないので、時間がかかるプロセスであるクエリーの書直しによって常に思ったとおりのパフォーマンスが得られるとは限りません。

Oracle Rdb バージョン 6.0 からは、クエリーに対するアウトラインを定義することで、オプティマイザがクエリーに対して選択するストラテジを直接制御できます。**クエリー・アウトライン**は、クエリーをどのように実装できるかに対する全体的な計画です。アウトラインは、クエリー・オプティマイザから作成されますが、抽出、編集、保管、使用または無視することができます。アウトラインは、クエリーを処理するときにオプティマイザが選択する結合順序、結合方法またはインデックスの使用方法（またはこれらのすべて）を制御するディレクティブを含むように定義できます。

アウトラインは、Oracle Rdb バージョン 6.0 以前のクエリー最適化プロセスに固有の安定性と制御可能性の問題を解決するために使用できます。以前のバージョンの Oracle Rdb からのアウトラインを保存および再利用することで、クエリーに対するバージョン間の安定性を得ることができます。アウトラインを編集することで、制御可能性を確保でき、オプティマイザが生成したソリューションを直接制御できます。場合によっては、オプティマイザが作成した結合順序、結合方法またはインデックス選択決定を手動で改善することで、実行時のパフォーマンスを向上させることができます。

アウトラインは、控えめに使用してください。オプティマイザが選択した検索ストラテジでは、ほとんどのクエリーでよいパフォーマンスが得られます。アウトラインは、オプティマイザが決定したストラテジでは実行状態を改善できない場合や、リリース間での安定性が重要な場合など、ごく少数の割合のクエリーについてのみ使用すべきものです。エンド・ユーザーやアプリケーション・プログラマーは、クエリーのアウトラインを定義すべきではありません。データベース管理者（DBA）、およびデータベースのパフォーマンス問題の検査に深い経験を持つ担当者が、クエリー・アウトラインの候補となるクエリーを検証する必要があります。場合によっては（特に新しいクエリーの場合）、クエリーのパフォーマンスが低いのは、オプティマイザが間違ったストラテジを選択しているからではなく、クエリーの記

述がよくないためです。このような場合、解決方法はアウトラインを定義することではなく、クエリーを書き直してより経済的にデータの検索を行うことです。

アウトラインが必要な場合、アウトラインは、DBA またはデータベース・パフォーマンス問題に深い経験を持つユーザーが定義する必要があります。オプティマイザは、利用可能な情報に基づいて検索ストラテジを選択します。場合によっては、データベース・パフォーマンスに深い経験を持つ要員のほうが、データの相互関係、データの配分、データのパーティションに関してオプティマイザよりも多くの知識を持っていることがあり、オプティマイザが選択するものより優れたストラテジを選択できることもあります。

部分アウトラインと完全アウトラインを指定できます。部分アウトラインでは、DBA がいくつかの重要な決定を行い、オプティマイザが残りの詳細を補足することができます。これによって、グローバルに最適なクエリー・プランを生成するための努力を最小化できます。完全アウトラインは、結合順序、結合方法およびクエリーに対するインデックスの使用方法を完全に制御できます。部分アウトラインと完全アウトラインの両方とも、オプティマイザが各クエリーに対して最適なプランの生成を保証する手段を提供します。ユーザーは、適宜クエリーを変更して、再ソートする必要はありません。5.9.4.2 項（5-66）では、完全アウトラインについて、5.9.4.3 項（5-66）では、部分アウトラインについて説明します。

同じクエリーに対して複数のアウトラインを作成することも可能です。これは、オプティマイザによって考慮されていない基準の処理にも便利です。たとえば、一日中実行しているクエリーを考えてください。日中、リソースの競合が高くなると予測される場合、リソースの競合がわずかまたはまったくない夜間に実行するクエリーとは異なる検索ソリューションが適切である場合があります。5.9.4.1 項（5-64）では、クエリーに対して複数のアウトラインを作成する方法を説明しています。

アウトラインは、大規模で重要なアプリケーションや非常に大きなデータベース（テラバイトサイズのデータベース、部分的に最適なクエリーの実行効果が拡大される場合）で最も有効です。

Oracle Rdb には、アウトラインの作成と編集を行う Windows インタフェースを持つ **Query Performance Tuner (QPT)** があります。QPT を使用しても、個々のクエリーを調整して、最大のパフォーマンスが得られます。QPT は、SQL クエリーの最適化ストラテジに対するグラフィカルなモデルを作成して、ソリューションを様々な角度（結合順序、アクセス・パス、結合方法、実行ストラテジ）から変更できるようにします。その後、このストラテジはデータベースに保存され、これ以降のクエリーのコンパイルと実行に適用されます。QPT の使用方法の詳細は、Windows のヘルプを参照してください。

5.9.1 項（5-53）から 5.9.9 項（5-79）では、アウトラインの作成、変更、削除および使用方法について説明します。

5.9.1 オプティマイザ出力を使用して保管するアウトラインを定義

Oracle Rdb オプティマイザによって生成されたクエリー・アウトラインを使用して、新しいアウトラインを定義します。

OpenVMS VAX OpenVMS Alpha

例 5-6（5-53）は、OpenVMS 上の RDMS$DEBUG_FLAGS 論理名と RDMS$DEBUG_FLAGS_OUTPUT 論理名を使用して、Oracle Rdb オプティマイザによって出力ファイルに生成されたアウトラインを獲得する方法を示しています。例の中のクエリーは、65 歳を超えたすべての従業員から得られた学歴情報を表示しています。

例 5-6 オプティマイザが生成したアウトラインの取得

```
$! Define RDMS$DEBUG_FLAGS to "Ss" so that outlines generated
$! by the optimizer are displayed.
$ DEFINE RDMS$DEBUG_FLAGS "Ss"
$!
$! Define RDMS$DEBUG_FLAGS_OUTPUT to be a file that will contain
$! the outlines generated during the session.
$ DEFINE RDMS$DEBUG_FLAGS_OUTPUT degrees_for_emps_over_65.sql
$ SQL
SQL> ATTACH 'FILENAME mf_personnel';
SQL> SELECT E.LAST_NAME, E.FIRST_NAME, E.EMPLOYEE_ID, D.DEGREE, D.YEAR_GIVEN
  1>   FROM EMPLOYEES E, DEGREES D
  2>   WHERE E.BIRTHDAY < '31-Dec-1928'
  3>         AND E.EMPLOYEE_ID = D.EMPLOYEE_ID
  4>   ORDER BY E.LAST_NAME;
 E.LAST_NAME      E.FIRST_NAME   E.EMPLOYEE_ID   D.DEGREE   D.YEAR_GIVEN
 Babbin           Joseph         00207          MA                 1980
 Babbin           Joseph         00207          MA                 1980
 Bartlett         Dean           00173          BA                 1978
 Clairmont        Rick           00231          BA                 1967
 Herbener         James          00471          BA                 1981
 Herbener         James          00471          MA                 1982
 Johnson          Bill           00240          BA                 1970
 Kinmonth         Louis          00177          BA                 1982
 Nash             Walter         00183          BA                 1977
 Nash             Walter         00183          PhD                1978
 O'Sullivan       Rick           00190          BA                 1982
 O'Sullivan       Rick           00190          MA                 1983
 Reitchel         Charles        00193          BA                 1982
 Ziemke           Al             00200          MA                 1971
 Ziemke           Al             00200          MA                 1971
15 rows selected
SQL> ROLLBACK;
SQL> DISCONNECT DEFAULT;
SQL> EXIT;
$!
$! Display the output file generated by the optimizer:
```

```
$ TYPE degrees_for_emps_over_65.sql
Leaf#01 FFirst RDB$RELATIONS Card=19
  BgrNdx1 RDB$REL_REL_NAME_NDX [1:1] Fan=8
-- Rdb Generated Outline :  8-JUN-1993 14:19(1)
create outline QO_982C2D52C1D95DA2_00000000
id '982C2D52C1D95DA2F46F0A7090B28309'
mode 0
as (
  query (
    subquery (
      RDB$RELATIONS 0         access path index       RDB$REL_REL_NAME_NDX
      )
    )
  )
compliance optional     ;
Sort
Cross block of 2 entries
  Cross block entry 1
    Leaf#01 BgrOnly RDB$RELATION_FIELDS Card=141
      BgrNdx1 RDB$RFR_REL_NAME_FLD_ID_NDX [1:1] Fan=8
  Cross block entry 2
    Get     Retrieval by index of relation RDB$FIELDS
      Index name  RDB$FIELDS_NAME_NDX [1:1]  Direct lookup
-- Rdb Generated Outline :  8-JUN-1993 14:19(2)
create outline QO_E8DB158FDFBD61CD_00000000
id 'E8DB158FDFBD61CDA331711179A010E6'
mode 0
as (
  query (
    subquery (
      RDB$RELATION_FIELDS 0   access path index
      RDB$RFR_REL_NAME_FLD_ID_NDX
        join by cross to
      RDB$FIELDS 1    access path index       RDB$FIELDS_NAME_NDX
      )
    )
  )
compliance optional     ;
Cross block of 2 entries
  Cross block entry 1
    Conjunct        Get     Retrieval by index of relation EMPLOYEES
      Index name  EMP_LAST_NAME [0:0]
  Cross block entry 2
    Leaf#01 FFirst DEGREES Card=165
      BgrNdx1 DEG_EMP_ID [1:1] Fan=17
-- Rdb Generated Outline :  8-JUN-1993 14:19(3)
create outline QO_284D6F269B44A56F_00000000
```

```
id '284D6F269B44A56F6C2BC8998832FD1D'
mode 0
as (
  query (
    subquery (
      EMPLOYEES 0     access path index       EMP_LAST_NAME
        join by cross to
      DEGREES 1       access path index       DEG_EMP_ID
      )
    )
  )
compliance optional     ;
```

◆

CompaqTru64 UNIX 上で RDB_DEBUG_FLAGS 構成パラメータと RDB_DEBUG_FLAGS_OUTPUT 構成パラメータを使用して、アウトラインを獲得することもできます。オプティマイザが生成したアウトラインを表示するには、RDMS$DEBUG_FLAGS 論理名または RDB_DEBUG_FLAGS 構成パラメータに "Ss"（大文字の S の後に小文字の s が続く）を指定する必要があることに注意してください。

例 5-6（5-53）では、出力ファイルには生成された 3 つのアウトラインが含まれています。各アウトラインの前に SQL コメント文字列、「- Rdb Generated Outline」と作成日時が入り、最後はセミコロン（;）で終了します。

次の番号は、例 5-6（5-53）の番号に対応しています。

1. オプティマイザが生成した最初のアウトラインは、オプティマイザがシステム・テーブルにアクセスするときに選択したストラテジを示しています。これは、オプティマイザが SELECT 文に対して生成したアウトラインではありません。
2. オプティマイザが生成した 2 番目のアウトラインも、システム・テーブルにアクセスするときに、オプティマイザが選択したストラテジを示しています。これは、オプティマイザが SELECT 文に対して生成したアウトラインではありません。
3. 3 番目のアウトラインは、オプティマイザが SELECT 文に対して生成したアウトラインです。このアウトラインは、オプティマイザがクエリーを処理するときに使用する EMPLOYEES テーブルと DEGREES テーブル、および EMP_LAST_NAME インデックスと DEG_EMP_ID インデックスを示しており、データベースに保管するために生成されたアウトラインです。

Oracle Rdb では、ビューはストアド・クエリーとして実装されます。クエリーがビューを使用する場合、そのクエリー用に生成されたアウトラインは、ビュー参照を、ビューを含むテーブルを示すサブクエリーに展開します。

OpenVMS VAX OpenVMS Alpha

例 5-7（5-56）は、オプティマイザが生成したアウトライン内でビューがどのように表されているかを示します。例 5-7（5-56）のクエリーは、CURRENT_JOB ビューと EMPLOYEES

テーブルを使用して、1982 年以来同じ仕事についている 65 歳を超えた従業員の情報を表示します。

例 5-7 オプティマイザが生成したアウトラインにおけるビューの表示

```
$! Define RDMS$DEBUG_FLAGS to "Ss" so that outlines generated by
$! the optimizer are displayed.
$ DEFINE RDMS$DEBUG_FLAGS "Ss"
$!
$! Define RDMS$DEBUG_FLAGS_OUTPUT to be a file that will contain
$! the outlines generated during the session.
$ DEFINE RDMS$DEBUG_FLAGS_OUTPUT over_65_10_years_in_job.sql
$ SQL
SQL> ATTACH 'FILENAME mf_personnel';
SQL> --
SQL> SELECT CJ.EMPLOYEE_ID, CJ.LAST_NAME, CJ.FIRST_NAME,
  1>   E.BIRTHDAY, CJ.JOB_START
  2>   FROM CURRENT_JOB CJ, EMPLOYEES E
  3>   WHERE CJ.EMPLOYEE_ID = E.EMPLOYEE_ID AND CJ.JOB_START > '31-Dec-1981'
  4>         AND E.BIRTHDAY < '31-Dec-1928';
 CJ.EMPLOYEE_ID   CJ.LAST_NAME     CJ.FIRST_NAME   E.BIRTHDAY    CJ.JOB_START
 00190            O'Sullivan       Rick            12-Jan-1923   25-Feb-1982
1 row selected
SQL> --
SQL> COMMIT;
SQL> DISCONNECT DEFAULT;
SQL> EXIT
$ DEASSIGN RDMS$DEBUG_FLAGS
$ DEFINE RDMS$DEBUG_FLAGS_OUTPUT
$!
$! Display the section of the over_65_10_years_in_job.sql output
$! file that was generated for the query:
$ TYPE over_65_10_years_in_job.sql
.
.
.
Cross block of 2 entries
  Cross block entry 1
    Conjunct        Get     Retrieval by index of relation EMPLOYEES
      Index name  EMP_EMPLOYEE_ID [0:0]
  Cross block entry 2
    Cross block of 2 entries
      Cross block entry 1
        Get     Retrieval by index of relation EMPLOYEES
          Index name  EMPLOYEES_HASH [1:1]       Direct lookup
      Cross block entry 2
        Conjunct        Conjunct        Get
        Retrieval by index of relation JOB_HISTORY
```

```
          Index name  JOB_HISTORY_HASH [1:1]
-- Rdb Generated Outline : 22-JUN-1993 16:22
create outline QO_B0B5EFA1486E0447_00000000
id 'B0B5EFA1486E0447C3B41C6E842558B6'
mode 0
as (
  query (
    subquery (
      EMPLOYEES 2     access path index       EMP_EMPLOYEE_ID
        join by cross to
      subquery (
        EMPLOYEES 1     access path index       EMPLOYEES_HASH (1)
          join by cross to
        JOB_HISTORY 0   access path index       JOB_HISTORY_HASH (2)
        )
      )
    )
  )
compliance optional     ;
$
```

◆

次の番号は、例 5-7（5-56）の番号に対応しています。

1. EMPLOYEES テーブルは、CURRENT_JOBS ビューの定義に使用されています。
2. JOB_HISTORY テーブルは、CURRENT_JOBS ビューの定義に使用されています。

オプティマイザが生成したアウトラインがすでに SQL CREATE OUTLINE 文形式に拡張されているので、テキスト・エディタを使用して出力ファイルから興味があるアウトラインを除く、すべてのテキストを削除できます。そして、編集された出力ファイルを SQL コマンド・プロシージャとして使用し、アウトラインを保存できます（出力ファイルはファイル・タイプ .sql であると仮定します）。例 5-8（5-57）では、3 番目に出力されたアウトラインを除くすべてのテキストが degrees_for_emps_over_65.sql ファイルから削除されています。出力ファイルを編集して無関係なテキストを取り除くときに、オプティマイザが生成したアウトライン名よりも意味があるアウトライン名を付けることもできます。たとえば、例 5-6（5-53）で 3 番目に生成されたアウトラインの名前を degrees_for_emps_over_65 に変更できます。出力ファイルを編集した結果を例 5-8（5-57）に示します。

例 5-8 オプティマイザが生成したアウトライン名の変更

```
-- Rdb Generated Outline :  8-JUN-1993 14:19
create outline DEGREES_FOR_EMPS_OVER_65 (1)
id '284D6F269B44A56F6C2BC8998832FD1D' (2)
mode 0
as (
  query (
    subquery (
```

```
      EMPLOYEES 0 (3)   access path index       EMP_LAST_NAME
        join by cross to
      DEGREES 1 (3)   access path index       DEG_EMP_ID
      )
    )
  )
compliance optional
comment is 'Created by Fred Smith' ; (4)
```

次の番号は、例 5-8（5-57）の番号に対応しています。

1. 生成されたアウトライン名（例 5-6（5-53）では、QO_284D6F269B44A56F_00000000）は意味がわかりやすいアウトライン名 degrees_for_emps_over_65 に変更されています。
2. ファイルを編集する場合にアウトライン ID を変更しないように注意してください。アウトライン ID を変更してしまうと、アウトラインと元のクエリーとの間の関連が壊れてしまい、それを保存すると、アウトラインは利用できなくなります。
3. オプティマイザがクエリーで使用するテーブルに対して生成したテーブル・インスタンス番号を変更しないでください。アウトラインでテーブルがたびたび現れる場合、アウトラインでのそのテーブルへの各参照に対して、一意なテーブル・インスタンス番号が生成されます。EMPLOYEES テーブルに対するテーブル・インスタンス番号は 0 です。DEGREES テーブルに対するテーブル・インスタンス番号は 1 です。
4. アウトラインに対してコメントを入れる場合には、COMMENT IS 句を追加します。

5.9.4 項（5-62）では、生成したアウトラインに行うことができる変更について説明しています。

データベースにアタッチして、SQL から degrees_for_emps_over_65.sql コマンド・ファイルを実行することでクエリーに対するアウトラインを保存します。保存された後、例 5-9（5-58）で示すように、アウトラインは SHOW OUTLINES 文で表示できます。

例 5-9 オプティマイザが生成したアウトラインの保存

```
SQL> ATTACH 'FILENAME mf_personnel';
SQL> @degrees_for_emps_over_65.sql
SQL> --
SQL> -- Display the outline using the SHOW OUTLINES statement:
SQL> SHOW OUTLINES degrees_for_emps_over_65
     DEGREES_FOR_EMPS_OVER_65
 Comment:       Created by Fred Smith
 Source:
create outline DEGREES_FOR_EMPS_OVER_65
id '284D6F269B44A56F6C2BC8998832FD1D'
mode 0
as (
  query (
    subquery (
```

```
      EMPLOYEES 0     access path index       EMP_LAST_NAME
        join by cross to
      DEGREES 1       access path index       DEG_EMP_ID
      )
    )
  )
compliance optional     ;
```

RDMS$DEBUG_FLAGS 論理名または RDB_DEBUG_FLAGS 構成パラメータを "Ss" として定義し、RDMS$DEBUG_FLAGS_OUTPUT 論理名または RDB_DEBUG_FLAGS_OUTPUT 構成パラメータを出力ファイルとして定義すると、オプティマイザが生成した各データベース・クエリーに対するアウトラインが出力ファイルに書き込まれます。オプティマイザは、Oracle Rdb データベースにクエリーを発行できる任意のフロントエンド製品（Oracle Rally [*1] など）に対してアウトラインを生成します。データベースにアクセスする階層構造の製品に対して生成するアウトラインは、SQL 文によって生成されるアウトラインと同じようにデータベースに保存できます。

ビューに対して保存されたアウトラインは、そのビューが他のクエリーに組み込まれている場合には適用できません。

オプティマイザが生成したアウトラインは、SQL CREATE OUTLINE 文形式に拡張されます。アウトラインを保存するには、アウトラインが参照するすべてのテーブルに対して SQL CREATE 権限を持っている必要があります。CREATE OUTLINE 文は、オンライン操作です（アウトラインが作成されたとき、他のユーザーはデータベースにアタッチすることができます）。

5.9.2 アウトライン・ディレクティブの指定

SQL CREATE OUTLINE 文を使用すると、クエリーを処理するときにオプティマイザが行う決定を制御するディレクティブを含むアウトラインを定義できます。次は、CREATE OUTLINE 文で指定できるアウトライン・ディレクティブの概要を示しています。これらのアウトラインに対する詳細な構文は、『Oracle Rdb7 SQL Reference Manual』の CREATE OUTLINE 文に説明されています。

- 結合順序
  結合順序は、最適化の実行時に、テーブル・インスタンス、ビュー・インスタンスまたはサブクエリー（結合項目）が互いに結合される順序です。アウトラインでは、次の項目を指定できます。
  - オプティマイザが、クエリーの実行中にアクセスした結合項目のグループに対して付ける順序
  - オプティマイザは、クエリーの実行中にアクセスした結合項目のグループに対して順序を付けない

[*1] OpenVMS システムのみ

- オプティマイザは、同じサブクエリー・レベル内で特定の結合項目に対して順序を付けない。オプティマイザが、同じレベルの結合項目を同じ順番でグループ化できる場合、これらの結合項目は移動可能であると言われています。

CREATE OUTLINE 構文を使用すると、結合項目を異なる順序で指定できますが、オプティマイザはアウトラインで指定されている結合順序を使用できない可能性があります。たとえば、外部コンテキストからの値が必要なサブクエリー内のコンテキストは、結合順序において外部コンテキストの前に置くことはできません。このような順序になると、オプティマイザはアウトラインで指定した結合ストラテジを使用できません。

- 結合方法
結合方法は、オプティマイザが結合においてレコードを関連付けるために使用するアルゴリズムです。アウトラインでは、オプティマイザが使用するクロス結合ストラテジ、一致結合ストラテジ、マージ・ストラテジまたは 2 つのデータ・ソースを結合する任意の方法を指定できます。
CREATE OUTLINE 文で指定できる構文的に有効な結合方法はいくつかありますが、オプティマイザはアウトラインで指定された結合方法を使用できない可能性があります。たとえば、次のような場合があります。
  - 一致結合ストラテジには、結合順序の内部コンテキストと外部コンテキストの間に等価結合列が存在することが必要です。アウトラインが作成されたクエリーが等価結合列を持っていなければ、オプティマイザはアウトラインで指定された一致結合ストラテジを使用できません。
  - マージ・ストラテジは、UNION オペレータを使用したクエリーに対してのみ有効です。そして、UNION オペレータを指定するすべてのクエリーは、マージ・ストラテジを使用する必要があります。
- アクセス・パス
アクセス・パスはテーブルの行の検索に使用する方法です。オプティマイザが返す行は、データベース・キー（dbkey）、順次読取り、インデックス（オプティマイザに使用させたいキーを指定）を使用して指定することも、またインデックスを使用せずに、または他のなんらかの方法によって指定することもできます。
オプティマイザが使用できないアクセス・パスを指定する、構文的に有効なアウトラインを作成できます。たとえば、dbkey が利用できないときに dbkey でアクセスを指定すると、オプティマイザはアウトラインが指定したアクセス・パスを使用できません。オプティマイザが使用できないアクセス・パスを指定した構文的に有効なアウトラインの例については、例 5-14（5-70）を参照してください。
- 準拠レベル
アウトラインは、必須または任意指定の準拠レベルを持つように定義できます。
アウトラインに対する準拠レベルが必須と定義されている場合、すべてのアウトライン・ディレクティブ（結合順序、結合方法およびインデックスの使用方法）は、アウトラインに従う必要があります。必須アウトラインの詳細は、5.9.4.4 項（5-67）を参照してください。
アウトラインに対する準拠レベルが任意指定であると定義されている場合、すべてのアウトライン・ディレクティブは任意指定です。オプティマイザは、任意指定の準拠アウ

トラインで指定されたすべてのディレクティブに従えない場合、クエリーを処理する別のストラテジを選択できます。任意指定のアウトラインの詳細は、5.9.4.5 項（5-70）を参照してください。

- 実行オプション
  オプティマイザが実行時に考慮しておく必要がある動的最適化オプションがあります。有効な実行オプションは、FAST FIRST、TOTAL TIME、ANY および NONE です。
  FAST FIRST 実行オプションは、オプティマイザが高速な最初の動的最適化を適切な場合に使用できることを示します。場合によっては、高速な最初の検索は使用できません。たとえば、クエリーで集計式（AVG、COUNT、MAX、MIN および SUM）が GROUP BY 句なしに使用されている場合、総時間検索を使用する必要があります。
  TOTAL TIME 実行オプションは、オプティマイザが総時間動的最適化を適切な場合に使用できることを示します。場合によっては、総時間検索は使用できません。たとえば、EXISTS 述語がクエリー内で使用されている場合、高速な最初の検索を使用する必要があります。
  ANY オプションは、オプティマイザが任意の最適化方法を自由に選択できることを示します。
  NONE オプションは、オプションの実行時最適化がまったく使用されないことを示します。NONE オプションを指定すると、どのバージョンの Oracle Rdb でも、クエリーは非常に安定して動作します。しかし、この安定性は、パフォーマンスを犠牲にして得られたようなものです。
  デフォルトの実行オプションは、ANY です。これは、動的最適化が有効であって、任意の動的最適化ストラテジが使用できるという意味です。

アウトライン・ディレクティブに対する構文の詳細は、『Oracle Rdb7 SQL Reference Manual』の CREATE OUTLINE 文の説明を参照してください。

5.9.3 ストアド・プロシージャのアウトラインの定義と保存

ストアド・プロシージャがデータベースに保管されたら、CREATE OUTLINE 文を使用して、そのストアド・プロシージャに対するアウトラインを定義して保管できます。『Oracle Rdb7 SQL プログラミングのガイド』には、ストアド・プロシージャをデータベース内に定義、保存する方法が説明されています。

例 5-10（5-61）は、ストアド・プロシージャに対するアウトラインを定義する ON PROCEDURE NAME 句を持つ SQL の CREATE OUTLINE 文の使用方法を示しています。

例 5-10 ストアド・プロシージャのアウトラインの定義

```
SQL> -- Store the procedure in the database:
SQL> CREATE MODULE my LANGUAGE sql AUTHORIZATION rick
  1> --
  2> PROCEDURE p1 (:x CHAR(14),  :y SMALLINT);
  3>    BEGIN
  4>    SELECT last_name INTO :x FROM employees LIMIT TO 1 ROW;
```

```
  5>    END;
  6> END MODULE;
SQL>
SQL> -- Create an outline for the new stored procedure:
SQL> CREATE OUTLINE LAST_NAME_OUTLINE ON PROCEDURE NAME P1
  1> MODE -1 COMPLIANCE OPTIONAL;
SQL> SHOW OUTLINES LAST_NAME_OUTLINE
     LAST_NAME_OUTLINE
 Source:
-- Rdb Generated Outline : 21-DEC-1993 09:15
create outline LAST_NAME_OUTLINE
id 'F7542B8932D2B8232131DCB52EAE205F'
mode -1
as (
  query (
    subquery (
      EMPLOYEES 0     access path index       EMP_LAST_NAME
      )
    )
  )
compliance optional     ;
SQL>
```

プロシージャ ID は、すでにデータベースに保管されている既存のストアド・プロシージャに対する有効なプロシージャ ID でなければなりません。CREATE OUTLINE 文の詳細な構文は、『Oracle Rdb7 SQL Reference Manual』を参照してください。

5.9.4 既存のアウトラインの変更

RMU Extract コマンドと SQL の CREATE OUTLINE 文を使用して、既存のアウトラインを変更できます。

OpenVMS VAX OpenVMS Alpha

例 5-11（5-62）に示すように RMU Extract コマンドを使用して、データベースに対するアウトラインを出力ファイルに抽出します。

例 5-11 既存のアウトラインをファイルに抽出

```
$ RMU/EXTRACT/ITEM=(OUTLINES)/LANGUAGE=SQL -
_$ /OUTPUT=degrees_for_emps_over_65.sql mf_personnel
$!
$! Show the RMU Extract output in degrees_for_emps_over_65.sql:
$ TYPE degrees_for_emps_over_65.sql
-- RMU/EXTRACT for Oracle Rdb V6.0-0                      8-JUN-1993 17:14:36.21
--
--                            Database Definition File
--
-- Source Database Name: SQL_DISK1:[RICK.V60]MF_PERSONNEL.RDB;1
```

```
------------------------------------------------------------------------------
set verify
set language ENGLISH
set default date format 'SQL92';
set quoting rules 'SQL92';
set date format DATE 001, TIME 001
attach 'pathname MF_PERSONNEL';
-- RMU/EXTRACT for Oracle Rdb V6.0-0                    8-JUN-1993 17:14:36.21
--
--                          Query Outline Definitions
--
------------------------------------------------------------------------------
create outline DEGREES_FOR_EMPS_OVER_65
id '284D6F269B44A56F6C2BC8998832FD1D'
mode 0
as (
    query (
        subquery (
            EMPLOYEES 0 access path index EMP_LAST_NAME
                join by cross to
            DEGREES 1 access path index DEG_EMP_ID
            )
        )
    )
compliance optional
execution options (any);
$
```

◆

例 5-11（5-62）で示したように RMU Extract コマンドはアウトラインを CREATE OUTLINE 文形式で抽出します。出力ファイル（この例では、degrees_for_emps_over_65.sql）を編集して、変更するアウトラインに無関係なテキストを削除し、アウトラインに望みの変更を加えます。

編集した出力ファイルに既存のアウトラインに望みの変更を加えた CREATE OUTLINE 文をが入っている場合、データベースにアタッチして既存のアウトラインを削除します（5.9.9 項（5-79）で示した SQL DROP OUTLINE 文を使用）。データベースに保存された各アウトラインは、一意な名前とアウトライン・モードを持つ必要があるので、既存のバージョンと同じアウトライン名とアウトライン・モードを持つアウトラインの変更バージョンは、既存のアウトラインを削除するまで保存できません。

既存のアウトラインを削除した後、データベースにアタッチして、変更した出力ファイルを SQL コマンド・プロシージャとして実行することで、変更したアウトラインを保存できます。

RMU Extract コマンドの詳細な説明は、『Oracle RMU Reference Manual』を参照してください。

5.9.4.1 1 つのクエリーに対して複数のアウトラインを作成

オプティマイザがクエリーに対するソリューションを生成するときに、すべての基準を考慮することはできないので、1 つのクエリーに対して 2 つ以上のアウトラインを定義したい場合があります。たとえば、オプティマイザがクエリーを処理するとき、リソースの競合が高いか低いかを決定できない場合があります。クエリーによっては、日中よりも夜間（リソースの競合は少ない場合）のほうが非常に速く実行されることも稀ではありません。この場合、DBA は、日中の処理に対して 1 つのアウトライン、そして夜間の処理に対して別のアウトラインを作成できます。

EMPLOYEES テーブルと DEGREES テーブルに対する競合が日中は高く、夜間は低ければ、夜間用として 2 番目のアウトラインを例 5-9（5-58）で定義したものの代替として定義できます。例 5-11（5-62）で示したように degrees_for_emps_over_65 アウトラインを出力ファイル degrees_for_emps_over_65.sql に抽出した後、抽出したアウトラインを変更して、同じクエリーで使用する別のアウトラインの定義を作成できます。

このクエリーが夜間に実行されるバッチ・ジョブの一部であれば、元の degrees_for_emps_over_65 アウトラインを変更して、夜間のクエリーのパフォーマンスを改良する総時間検索方法を得ることもできます。高速な最初の検索（オプティマイザが元の degrees_for_emps_over_65 アウトラインに対して選択した方法）は、クエリーを満足する最初の数レコードに対する検索時間を最適化します。クエリーが対話的に実行され、クエリーを満足するすべてのレコードが返される前に終了する（たとえば、ユーザーが [Ctrl] キーと [C] キーを押す場合）場合には、高速な最初の最適化は優れた選択です。

クエリーが夜間にバッチ・ジョブの一部として実行される場合、このクエリーはすべてのレコードが検索されるまで停止しません。総時間検索方法は、総検索時間を最適化するので、このクエリーにとって夜間に実行するのが最も適切な検索方法と言えます。この変更は、抽出されたアウトラインの "execution options (any)" を "execution options (total time)" に変更することで行えます。検索方法の指定の説明は、5.9.2 項（5-59）を参照してください。

1 つ以上のアウトラインがデータベースに保存されているクエリーに対してアウトラインを定義する場合、定義するアウトラインは一意なアウトライン名とアウトライン・モードを持つ必要があります。例 5-11（5-62）に示したように、この例の既存のアウトラインは、degrees_for_emps_over_65 というアウトライン名で、デフォルトのアウトライン・モードは 0 です。アウトライン・モードは符号付き整数で、有効な値の範囲は -2,147,483,648 ～ 2,147,483,647 です。クエリーに対する新しいアウトラインは、例 5-12（5-65）で示すように、アウトライン名 degrees_for_emps_over_65_night で、アウトライン・モードは -1 です。正のモード値は、オラクル社が将来使用するために予約しています。したがって、モード値には、0 ～ -2,147,483,648 までの値を指定するようお薦めします。アウトラインに指定するアウトライン・モード値は、同時に使用される他のアウトラインと共有する値であることが必要です。たとえば、夜間のみ実行されるアウトラインに対して、アウトライン・モード値を -1 にするように決定することもできます。例 5-12（5-65）では、degrees_for_emps_over_65_night アウトラインには、この理由から -1 のアウトライン・モード値が指定されています。

すべての変更を行った後では、変更された出力ファイル degrees_for_emps_over_65.sql は、次のようになります。

```
create outline DEGREES_FOR_EMPS_OVER_65_NIGHT
id '284D6F269B44A56F6C2BC8998832FD1D'
mode -1
as (
    query (
        subquery (
            EMPLOYEES 0 access path index EMP_LAST_NAME
                join by cross to
            DEGREES 1 access path index DEG_EMP_ID
            )
        )
    )
compliance optional
execution options (TOTAL TIME);
```

例 5-12（5-65）に示すように、degrees_for_emps_over_65.sql ファイルはアウトラインを保存するために使用されます。

例 5-12 クエリーに対して複数のアウトラインを作成

```
SQL> --
SQL> -- The output file is used as an SQL command procedure to define
SQL> -- and store the outline:
SQL> @DEGREES_FOR_EMPS_OVER_65.SQL
SQL> --
SQL> -- Display the outline:
SQL> SHOW OUTLINES DEGREES_FOR_EMPS_OVER_65_NIGHT
     DEGREES_FOR_EMPS_OVER_65_NIGHT
 Source:
create outline DEGREES_FOR_EMPS_OVER_65_NIGHT
id '284D6F269B44A56F6C2BC8998832FD1D'
mode -1
as (
  query (
    subquery (
      EMPLOYEES 0     access path index       EMP_LAST_NAME
        join by cross to
      DEGREES 1       access path index       DEG_EMP_ID
      )
    )
  )
compliance optional
execution options (      total time )   ;
```

アウトラインは、次の項目に対してマルチオクテット・キャラクタをサポートしています。

- アウトライン名
- アウトラインで指定されたインデックス名とテーブル名
- アウトラインで指定された説明とコメント

『Oracle Rdb7 SQL Reference Manual』のキャラクタ・セットの項にマルチオクテット・キャラクタの詳細な説明があります。

5.9.4.2 完全アウトライン

完全アウトラインでは、アウトライン・ディレクティブは、クエリー内のすべてのテーブルと結合の組合せに対して、オプティマイザが使用する必要があるストラテジに関するすべての内容を指定します。つまり、完全アウトラインは、5.9.2 項（5-59）で説明した各ディレクティブを指定します。

オプティマイザが生成したアウトラインは、常に完全アウトラインです。

完全アウトラインは、Oracle Rdb のあるバージョンから別のバージョンに移行した場合にも、オプティマイザに対して変更が行われているにもかかわらず、クエリーのパフォーマンスが安定したままであることを保証する場合に特に便利です。完全アウトラインでは、オプティマイザは常にアウトラインが指定した結合順序、アクセス・パス、結合方法および実行オプションを選択するように制約されています。

オプティマイザは、アウトラインで指定されているすべてのテーブルとインデックスがデータベース内に存在する場合のみ、完全アウトライン内のすべてのディレクティブに従うことができます。完全アウトラインで指定されたインデックスやテーブルが削除された場合には、Oracle Rdb は、アウトラインを無効としてマークします。そして、オプティマイザは、クエリーの処理時にアウトライン内のすべてのディレクティブに従うことはできません。無効なアウトラインの説明は、5.9.8 項（5-78）を参照してください。

5.9.4.3 部分アウトライン

部分アウトラインでは、アウトライン・ディレクティブは、クエリー内のすべてのテーブルと結合の組合せに対してオプティマイザが使用するストラテジに関して、すべての内容を指定するわけではありません。つまり、部分アウトラインは、5.9.2 項（5-59）で説明した個々のディレクティブを指定しません。

部分アウトラインでは、オプティマイザにある程度の自由があるため、代替手段を選択することができます。これは、完全アウトラインと比較して、部分アウトラインは無効になる可能性が小さいことを示しています。次の場合、アウトラインは部分的です。

- ANY オプションが、結合方法、アクセス・パスまたは実行オプションに対して指定されている
- FLOATING 句または UNORDERED 句がアウトライン定義のどこかで使用されている
- クエリーの一部であるサブクエリー、結合またはテーブルがアウトラインから省かれている

場合によっては、部分アウトラインのほうが完全アウトラインよりも優れていることもあります。部分アウトラインでは、オプティマイザが一緒に保持すべき結合項目のコレクションを正しく指定すること、およびこれらの結合項目を順序付けるか、順序付けないかを正しく指定することが最も重要です。ANY の結合方法や ANY のアクセス・パスを指定すること、およびソリューションにとって重要でないテーブルを省略することで、オプティマイザを過剰に制約するのを防止できます。これによって、オプティマイザは、データ配分と実行時変数の結合における変更を調整できます。

クエリー用にアウトラインを定義する場合、段階的にアウトラインを改良できます。最初のステップは、オプティマイザが詳細を理解できるように、結合順序の先頭を制約することです。この最初のステップによってクエリーの実行時パフォーマンスが容認できる程度に改善されない場合には、こアウトラインを保存した後、アウトラインに対してさらに改良を加えることができます。

部分アウトラインでは、重要でないアウトライン要素は指定されていません。重要でないテーブルやインデックスへの参照を削除することで完全アウトラインを変更すると、結果として得られる部分アウトラインでは、以前の完全アウトラインと比較して、特定のテーブルやインデックスに対する依存度が低くなります。つまり、データベースから重要でないインデックスやテーブルを削除しても、新しい部分アウトラインはこの削除によって無効になることはありません。しかし、完全アウトラインと同様に、部分アウトラインで指定されたテーブルやインデックスを削除すると、Oracle Rdb は、このアウトラインを無効なものとしてマークし、クエリーの処理時には使用できなくなります。無効なアウトラインの説明は、5.9.8 項（5-78）を参照してください。

アウトライン・ディレクティブの指定に関する詳細な構文は、『Oracle Rdb7 SQL Reference Manual』の CREATE OUTLINE 文の説明を参照してください。

5.9.4.4 必須アウトライン

必須準拠レベルで定義されているアウトラインでは、オプティマイザは、結合順序、結合方法およびインデックスの使用方法など、すべてのアウトライン・ディレクティブに従う必要があります。オプティマイザが必須アウトライン内のすべてのアウトライン・ディレクティブに従うことができない場合には、クエリーは失敗します。

OpenVMS VAX OpenVMS Alpha

例 5-13（5-67）は、クエリーが失敗したときに表示されるエラー・メッセージを示しています。

例 5-13 オプティマイザが必須アウトラインのディレクティブに従うことができない場合に表示されるエラー・メッセージ

```
$! Define the appropriate logical names so that outlines generated
$! by the optimizer can be logged to an output file.
$ DEFINE RDMS$DEBUG_FLAGS "Ss"
$ DEFINE RDMS$DEBUG_FLAGS_OUTPUT employees_outline.sql
$ SQL$
SQL> ATTACH 'FILENAME mf_personnel';
SQL> --
SQL> -- Execute a query for which an outline will be defined.
```

```
SQL> SELECT EMPLOYEE_ID FROM EMPLOYEES WHERE EMPLOYEE_ID BETWEEN
  1> '00150' AND '00175';
 EMPLOYEE_ID
 00164
 00165
 00166
 00167
 00168
 00169
 00170
 00171
 00172
 00173
 00174
 00175
12 rows selected
SQL> COMMIT;
SQL> DISCONNECT DEFAULT;
SQL> EXIT;
$ DEASSIGN RDMS$DEBUG_FLAGS_OUTPUT
$!
$! Display the portion of the generated optimizer output for
$! the query for which the outline will be defined.  The output
$! shows that the optimizer used the sorted index EMP_EMPLOYEE_ID
$! to retrieve the range of EMPLOYEE_ID column values specified by
$! the SELECT query.
$ TYPE employees_outline.sql
   .
   .
   .
--  Rdb Generated Outline : 11-JUN-1993 12:15
create outline QO_FAF0A23CF87C0FEE_00000000
id 'FAF0A23CF87C0FEE840D9BCA37A236B5'
mode 0
as (
  query (
    subquery (
      EMPLOYEES 0     access path index       EMP_EMPLOYEE_ID
      )
    )
  )
compliance optional     ;
$!
$! Edit the generated outline, changing the generated outline name
$! for the query to FIND_EMPLOYEE_IDS, the index used to EMPLOYEES_HASH,
$! and the compliance level to mandatory.
$ EDIT employees_outline.sql
```

```
   .
   .
   .
$!
$! Display the modified output file.
$ TYPE employees_outline.sql
--  Rdb Generated Outline : 11-JUN-1993 12:15
create outline FIND_EMPLOYEE_IDS
id 'FAF0A23CF87C0FEE840D9BCA37A236B5'
mode 0
as (
  query (
    subquery (
      EMPLOYEES 0     access path index       EMPLOYEES_HASH
      )
    )
  )
compliance mandatory     ;
$ DEASSIGN RDMS$DEBUG_FLAGS
$ !
$ ! Define and store the outline, using the modified output file as an
$ ! SQL command procedure:
$ SQL$
SQL> ATTACH 'FILENAME mf_personnel';
SQL> @employees_outline.sql
SQL> SHOW OUTLINES FIND_EMPLOYEE_IDS
     FIND_EMPLOYEE_IDS
 Source:
create outline FIND_EMPLOYEE_IDS
id 'FAF0A23CF87C0FEE840D9BCA37A236B5'
mode 0
as (
  query (
    subquery (
      EMPLOYEES 0     access path index       EMPLOYEES_HASH
      )
    )
  )
compliance mandatory    ;
SQL> COMMIT;
SQL> --
SQL> -- When the query is issued now, it fails because the outline
SQL> -- directives specified as mandatory cannot be followed (the
SQL> -- hashed index EMPLOYEES_HASH cannot be used for the range
SQL> -- retrieval of EMPLOYEE_ID column values required by the query).
SQL> SELECT EMPLOYEE_ID FROM EMPLOYEES WHERE EMPLOYEE_ID BETWEEN
  1> '00150' AND '00175';
```

```
%RDMS-F-OUTLINE_FAILED, could not comply with mandatory query outline directives
SQL>
```

◆

クエリーに対する優先順位として、クエリーの処理時には特定のオプティマイザ・ストラテジを常に使用したい場合には、アウトラインに対して必須準拠レベルを指定するのが適切であると考えられます。オプティマイザがアウトライン・ディレクティブのすべてについては従うことができない場合には、クエリーは失敗します。このことは、アウトラインが無効になった理由を探索する必要があります。

5.9.4.5 任意指定アウトライン

任意指定準拠レベルで定義されているアウトラインの場合、オプティマイザは、すべてのアウトライン・ディレクティブに従う必要はありません。オプティマイザは、任意指定の準拠アウトラインで指定されたすべてのディレクティブに従うことができない場合には、ストラテジを選択してクエリーを処理できます。

例 5-14（5-70）は、オプティマイザが任意指定の準拠レベルで定義されたアウトラインのアウトライン・ディレクティブの 1 つに従えない場合、オプティマイザは代替ストラテジを選択してクエリーを処理します。

OpenVMS VAX OpenVMS Alpha

例 5-14（5-70）では、任意指定の準拠レベルを持つアウトラインの場合、RDMS$DEBUG_FLAGS 論理名を "Ss" と定義した場合のみ、Oracle Rdb は、アウトライン・ディレクティブが完全にはコンパイルされなかった場合を通知します。

例 5-14 任意指定の準拠レベルアウトラインが完全にコンパイルされたかどうかの判断

```
$! The RDMS$DEBUG_FLAGS logical name is not set to "Ss".
$ SHOW LOGICAL RDMS$DEBUG_FLAGS
%SHOW-S-NOTRAN, no translation for logical name RDMS$DEBUG_FLAGS
$ SQL
SQL> ATTACH 'FILENAME mf_personnel';
SQL> --
SQL> -- Note that the compliance level for the outline is optional
SQL> -- in this example.
SQL> SHOW OUTLINES FIND_EMPLOYEE_IDS
     FIND_EMPLOYEE_IDS
 Source:
create outline FIND_EMPLOYEE_IDS
id 'FAF0A23CF87C0FEE840D9BCA37A236B5'
mode 0
as (
  query (
    subquery (
      EMPLOYEES 0     access path index       EMPLOYEES_HASH
      )
    )
  )
```

```
compliance optional     ;
SQL> --
SQL> -- Issue the query for which the outline was defined.  When the
SQL> -- RDMS$DEBUG_FLAGS logical name is not set to "Ss", Oracle Rdb
SQL> -- gives no indication that the directives in the outline
SQL> -- were not fully complied with.
SQL> SELECT EMPLOYEE_ID FROM EMPLOYEES WHERE EMPLOYEE_ID BETWEEN
  1> '00150' AND '00175';
 EMPLOYEE_ID
 00166
 00167
 00173
 00169
 00175
 00172
 00164
 00168
 00165
 00171
 00170
 00174
12 rows selected
SQL> COMMIT;
SQL> DISCONNECT DEFAULT;
SQL> EXIT;
$!
$! Define RDMS$DEBUG_FLAGS as "Ss".
$ DEFINE RDMS$DEBUG_FLAGS "Ss"
$ SQL
SQL> ATTACH 'FILENAME mf_personnel';
SQL> --
SQL> -- Issue the original query again.  Notice that with the
SQL> -- RDMS$DEBUG_FLAGS logical name set to "Ss", Oracle Rdb informs
SQL> -- you that the FIND_EMPLOYEE_IDS outline was used, but full
SQL> -- compliance with the outline was not possible.  The display
SQL> -- also shows that the optimizer used sequential retrieval to
SQL> -- return the requested EMPLOYEE_ID column values (because
SQL> -- the EMPLOYEES_HASH index specified in the outline is not
SQL> -- able to retrieve a range of values).
SQL> SELECT EMPLOYEE_ID FROM EMPLOYEES WHERE EMPLOYEE_ID BETWEEN
  1> '00150' AND '00175';
   .
   .
   .
~S: Outline FIND_EMPLOYEE_IDS used
~S: Full compliance with the outline was not possible
Conjunct        Get     Retrieval sequentially of relation EMPLOYEES
```

```
--  Rdb Generated Outline : 11-JUN-1993 13:04
create outline QO_FAF0A23CF87C0FEE_00000000
id 'FAF0A23CF87C0FEE840D9BCA37A236B5'
mode 0
as (
  query (
    subquery (
      EMPLOYEES 0     access path sequential
      )
    )
  )
compliance optional     ;
 EMPLOYEE_ID
 00166
 00167
 00173
 00169
 00175
 00172
 00164
 00168
 00165
 00171
 00170
 00174
12 rows selected
SQL>
```

◆

例 5-14（5-70）は、ユーザーからオプティマイザのパフォーマンスが低いことにクレームが出た場合、DBA は、オプティマイザがストラテジを選択した結果なのか、アウトラインに従った結果なのかを判断する必要があります。アウトラインが使用されている場合、クエリーに対して出力されたストラテジには、"~S Outline outline-name used" の文字列が含まれます。

クエリーが常に成功することが最も優先される場合には、任意指定準拠レベルを指定するのは適切です。オプティマイザは、任意指定アウトラインのすべてのアウトライン・ディレクティブに従うことができない場合、クエリーを処理する別のストラテジを選択できます。

5.9.5 OPTIMIZE 句を使用してクエリー用のアウトラインを選択

Oracle Rdb は、実行される各クエリーに対してアウトラインとアウトライン ID を生成します。アウトラインはオプティマイザが生成し、アウトライン ID は、クエリー全体のコンパイルに基づきます。クエリーがコンパイルされて、クエリーに対するアウトライン ID が生成されると、Oracle Rdb は、クエリーと同じアウトライン ID を持つアウトラインを探します。Oracle Rdb は、クエリーと同じアウトライン ID を持つアウトラインを見つけたら、そのアウトライン内のディレクティブを使用してクエリーを実行します。
RDMS$DEBUG_FLAGS 論理名または RDB_DEBUG_FLAGS 構成パラメータを "Ss" と定義した場合、Oracle Rdb は、実行する各クエリーに関するアウトラインとアウトライン ID を表示します。

SQL は、クエリーが対話的に実行される場合とプログラムで実行される場合で、同じクエリーをわずかに異なる方法でコンパイルします。この場合、同じクエリーでも、対話型クエリー用に生成されたアウトライン ID と、プログラム用に生成されたアウトライン ID では異なります。このシナリオでは、対話型クエリー用のアウトラインでは、クエリーとアウトラインが異なるアウトライン ID を持っているため、Oracle Rdb は、プログラムではクエリー用のアウトラインを自動的には使用しません。

同様に、異なるバージョンの Oracle Rdb では、SQL が変更されているため、クエリーは異なる方法でコンパイルされることがあります。このような場合、あるバージョンの Oracle Rdb を使用してクエリー用に生成されたアウトライン ID は、別のバージョンの Oracle Rdb を使用してクエリー用に生成されたアウトライン ID とは異なります。このシナリオでは、1 つのバージョンで作成されたクエリーに対するアウトラインが存在する場合、クエリーとアウトラインが異なるアウトライン ID を持っているので、クエリーが異なるバージョンで実行されている場合、Oracle Rdb はアウトラインを自動的には使用しません。

Oracle Rdb バージョン 6.1 以降では、SELECT 文の OPTIMIZE USING 句を使用すると、クエリーで使用するアウトラインを明示的に指定できます。これは、特定のアウトラインをクエリーで使用するように指定できること、そしてクエリーとアウトラインのアウトライン ID が異なっていても、Oracle Rdb は指定したアウトラインを使用することを意味しています。

たとえば、女性の従業員が取得した学位を検出する次のクエリーに対して、WOMENS_DEGREES というアウトラインを作成したとします。

```
SQL> SELECT E.LAST_NAME, E.EMPLOYEE_ID, D.DEGREE, D.DEGREE_FIELD, D.YEAR_GIVEN
  1>   FROM EMPLOYEES E, DEGREES D
  2>   WHERE E.SEX = 'F' AND E.EMPLOYEE_ID = D.EMPLOYEE_ID
  3>   ORDER BY LAST_NAME
```

次の SHOW OUTLINE 文は、Oracle Rdb がクエリー用に生成したアウトラインとアウトライン ID を使用して以前に作成したアウトラインを表示します。

```
SQL> SHOW OUTLINE WOMENS_DEGREES
     WOMENS_DEGREES
 Source:
```

```
create outline WOMENS_DEGREES
id 'D3A5BC351F507FED820EB704FC3F61E8'  <-- 元のクエリーのアウトライン ID
mode 0
as (
  query (
    subquery (
      EMPLOYEES 0     access path index       EMP_EMPLOYEE_ID
        join by cross to
      DEGREES 1       access path index       DEG_EMP_ID
      )
    )
  )
compliance optional;
SQL>
```

元のクエリーに似たクエリー（この例では、LIMIT TO オペレータが唯一の変更点）を指定すると、クエリーのコンパイルも変化する可能性があります。RDMS$DEBUG_FLAGS 論理名または RDB_DEBUG_FLAGS 構成パラメータを "Ss" と定義すると、次の出力に示すように、Oracle Rdb は新しいクエリーに対して異なるアウトライン ID を生成します。

```
SQL> SELECT E.LAST_NAME, E.EMPLOYEE_ID, D.DEGREE, D.DEGREE_FIELD, D.YEAR_GIVEN
  1>    FROM EMPLOYEES E, DEGREES D
  2>    WHERE E.SEX = 'F' AND E.EMPLOYEE_ID = D.EMPLOYEE_ID
  3>    ORDER BY LAST_NAME
  4>    LIMIT TO 10 ROWS;
   .
   .
   .
--  Rdb Generated Outline : 21-JUN-1994 15:41
create outline QO_74C62CA1A8532543_00000000
id '74C62CA1A8532543D57668C8F5BCDB92'  <--- 異なるアウトライン ID
mode 0
as (
  query (
    subquery (
      EMPLOYEES 0     access path index       EMP_EMPLOYEE_ID
        join by cross to
      DEGREES 1       access path index       DEG_EMP_ID
      )
    )
  )
compliance optional;
 E.LAST_NAME      E.EMPLOYEE_ID   D.DEGREE   D.DEGREE_FIELD    D.YEAR_GIVEN
 Boyd             00244           MA         Elect. Engrg.             1982
   .
   .
   .
```

```
 Clinton          00201           MA         Applied Math              1978
10 rows selected
SQL>
```

オプティマイザは、元のクエリーに対して行ったように、新しいクエリーに対して同じストラテジを選択します。しかし、Oracle Rdb は新しいクエリーを元のクエリーとは異なる方法でコンパイルし、異なるアウトライン ID を生成します。

OPTIMIZE USING 句を使用して WOMENS_DEGREES アウトラインを指定することで、Oracle Rdb が WOMENS_DEGREES アウトラインを使用してクエリーを実行しようとしていることを確認できます。

```
SQL> SELECT E.LAST_NAME, E.EMPLOYEE_ID, D.DEGREE, D.DEGREE_FIELD, D.YEAR_GIVEN
  1>    FROM EMPLOYEES E, DEGREES D
  2>    WHERE E.SEX = 'F' AND E.EMPLOYEE_ID = D.EMPLOYEE_ID
  3>    ORDER BY LAST_NAME
  4>    LIMIT TO 10 ROWS
  5>    OPTIMIZE USING WOMENS_DEGREES;
~S: Outline WOMENS_DEGREES used  <-- クエリーはWOMENS_DEGREESアウトラインを使用
   .
   .
   .
 E.LAST_NAME      E.EMPLOYEE_ID   D.DEGREE   D.DEGREE_FIELD    D.YEAR_GIVEN
 Boyd             00244           MA         Elect. Engrg.             1982
   .
   .
   .
 Clinton          00201           MA         Applied Math              1978
10 rows selected
SQL>
```

この例の出力は、Oracle Rdb が WOMENS_DEGREES アウトラインを使用してクエリーを実行したことを示しています。

Oracle Rdb は、アウトライン内にあるすべてのディレクティブに従うことができるならば、OPTIMIZE USING 句で指定したアウトラインを使用します（たとえば、アウトラインの準拠レベルが必須であり、アウトライン・ディレクティブで指定されているインデックスの 1 つが削除された場合には、アウトラインは使用されず、エラー・メッセージが表示されます）。

存在しないアウトライン名を指定すると、Oracle Rdb はクエリーをコンパイルして、指定したアウトライン名を無視し、クエリーと同じアウトライン ID を持つ既存のアウトラインを探します。同じアウトライン ID のアウトラインを見つけたら、そのアウトライン内のディレクティブを使用してクエリーを実行してみます。クエリーと同じアウトライン ID を持つアウトラインが見つからなければ、オプティマイザはそのクエリーに対するストラテジを選択し、クエリーの実行にそのストラテジを使用します。

クエリー用に定義されていないアウトラインの名前を指定すると、Oracle Rdb は指定したアウトラインを使用しようとしますが、アウトラインを正しく使用できなければエラー・メッセージを返します。

OPTIMIZE AS 句を使用してクエリーの名前を指定することもできます。RDMS$DEBUG_FLAGS 論理名または RDB_DEBUG_FLAGS 構成パラメータを "Ss" として定義した場合、Oracle Rdb は、このクエリー用に生成したアウトライン内でクエリー名をアウトライン名として使用します。たとえば、次のように行います。

```
SQL> ATTACH 'FILENAME mf_personnel';
   .
   .
   .
SQL> --  Specify the query name WOMENS_DEGREES for the query:
SQL> SELECT E.LAST_NAME, E.EMPLOYEE_ID, D.DEGREE, D.DEGREE_FIELD, D.YEAR_GIVEN
  1>     FROM EMPLOYEES E, DEGREES D
  2>     WHERE E.SEX = 'F' AND E.EMPLOYEE_ID = D.EMPLOYEE_ID
  3>     ORDER BY LAST_NAME
  4>     OPTIMIZE AS WOMENS_DEGREES;
~Query Name : WOMENS_DEGREES        <---- クエリー名を表示
   .
   .
   .
create outline WOMENS_DEGREES        <---- クエリー名をアウトライン名として使用
   .
   .
   .
compliance optional;
 E.LAST_NAME      E.EMPLOYEE_ID   D.DEGREE   D.DEGREE_FIELD    D.YEAR_GIVEN
 Boyd             00244           MA         Elect. Engrg.             1982
   .
   .
   .
 Watters          00186           BA         Arts                      1975
61 rows selected
SQL>
```

選択式における OPTIMIZE USING 句と OPTIMIZE AS 句の指定の説明は、『Oracle Rdb7 SQL Reference Manual』を参照してください。

5.9.6 論理名を使用してオプティマイザが使用するアウトラインを制御

5.9.4.1 項（5-64）では、クエリーに対して複数のアウトラインを作成する方法を説明しています。クエリーに対して複数のアウトラインが存在する場合、RDMS$BIND_OUTLINE_MODE 論理名か RDB_BIND_OUTLINE_MODE 構成パラメータに、オプティマイザが使用するアウトラインに対するアウトライン・モードの値を設定する必要があります。

たとえば、あるクエリーに対して 2 つのアウトラインが保存されているとします。1 つのアウトラインがデフォルトのアウトライン・モード値である 0 を持っており、他のアウトラインが -1 のアウトライン・モード値を持っているとします。オプティマイザにアウトライン・モード値 0 のアウトラインを使用させる場合、RDMS$BIND_OUTLINE_MODE 論理名か RDB_BIND_OUTLINE_MODE 構成パラメータに 0（ゼロ）を設定する必要があります。オプティマイザにクエリー用のもう 1 つのアウトラインを使用させる場合、RDMS$BIND_OUTLINE_MODE 論理名か RDB_BIND_OUTLINE_MODE 構成パラメータに -1 を設定する必要があります。

あるクエリーに対して保存されたアウトラインをオプティマイザに無視させる場合、RDMS$BIND_OUTLINE_FLAGS 論理名または RDB_BIND_OUTLINE_FLAGS 構成パラメータの値を I と定義します。あるプロセスが RDMS$BIND_OUTLINE_FLAGS 論理名または RDB_BIND_OUTLINE_FLAGS 構成パラメータを I として定義された場合、クエリーの処理時にオプティマイザは、保存されたアウトラインを無視します。

表 5-3（5-77）に、設定すべき値を示します。

表 5-3 クエリーで使用するアウトラインを指定する論理名と構成パラメータ値

論理名 構成パラメータ	値	内容
RDMS$BIND_OUTLINE_FLAGS RDB_BIND_OUTLINE_FLAGS	「I」	存在する場合には、アウトラインを無視
RDMS$BIND_OUTLINE_MODE RDB_BIND_OUTLINE_MODE	有効なモード	選択すべきアウトラインのモードを指定

5.9.7 ユーザーにとってアウトラインの目に見える効果

ユーザーは、次の場合に限り、アウトラインがデータベースに保存されていることがわかります。

- SHOW OUTLINES 文や RMU Extract コマンドを使用して、格納されたアウトラインを表示または抽出
- RDMS$DEBUG_FLAGS か RDB_DEBUG_FLAGS に "Ss" を設定して、クエリーに対してアウトラインが使用されたことをストラテジ・ダンプで通知

- 発行したクエリーに対する必須アウトラインのコンパイルを完了できないため、クエリーが失敗し、Oracle Rdb からのエラー・メッセージを受信

5.9.8 格納されたアウトラインの無効化

データベース内で行われた様々な変更（メタデータの変更など）によって、データベースに格納されたアウトラインを使用できなくなることがあります。たとえば、アウトラインで指定されているインデックスやテーブルの削除をユーザーがコミットすると、このアウトラインは無効としてマークされます。オプティマイザは、無効になったアウトラインを、クエリーを処理する際のアウトラインの候補とは見なしません。

データベース内に格納された無効なアウトライン名を判断する方法が 2 通りあります。

- SHOW OUTLINES 文
  あるアウトラインが無効かどうかを確認するには、例 5-15（5-78）に示すように SHOW OUTLINES 文を使用してアウトライン名を指定します。

  例 5-15 特定のアウトラインの妥当性を確認するために SHOW OUTLINES 文を使用

```
SQL> SHOW OUTLINES NEW_JOB_STARTS_SINCE_1982
     NEW_JOB_STARTS_SINCE_1982
*** Query outline marked as invalid ***
 Source:
create outline NEW_JOB_STARTS_SINCE_1982
id '96231ABDD4A30FC73ABE782B14762028'
mode 0
as (
  query (
    subquery (
      EMPLOYEES 0     access path index       EMP_EMPLOYEE_ID
        join by cross to
      JOB_HISTORY 1   access path index       JOB_HISTORY_HASH
      )
    )
  )
compliance optional;
SQL>
```

  特定のアウトラインの名前を指定せずに SHOW OUTLINES 文を発行すると、データベースに保存されているすべてのアウトラインの名前が表示されますが、無効なアウトラインは無効としてマークされていません。

- RMU Extract コマンド
  格納されたすべてのアウトラインのうち、どれが無効か確認したい場合には、次のように行ってください。

1. RMU Extract コマンドを使用して、格納されたすべてのアウトラインをファイルに抽出します。

```
$ RMU/EXTRACT/ITEM=OUTLINES/OUTPUT=extracted-outlines.log db-name
$ rmu -extract -item=outlines -output=extracted-outlines.log db-name
```

2. テキスト・エディタを使用して、RMU Extract コマンドで得られた出力ファイル内で文字列 "invalid" を検索します。無効なアウトラインの名前を記録します。

格納されたアウトラインは、無効になった後はもう一度有効にすることはできません。そのかわり、DROP OUTLINE 文を使用して無効なアウトラインを削除し、新しいアウトラインを作成できます。次のステップを行うと、無効なアウトラインの新しいバージョンを容易に作成できます。

1. 無効なアウトラインを出力ファイルに抽出します。
2. テキスト・エディタを使用して、出力ファイル内に抽出したアウトラインに対して必要な変更を行います。
3. DROP OUTLINE 文を使用して、無効なアウトラインを削除します。
4. 出力ファイルを SQL プロシージャとして使用して、新しいバージョンのアウトラインを保存します。

格納された各アウトラインについて、アウトライン用のストレージ、およびアウトラインを通して行う能率的な検索に使用するインデックス（群）のために、数百バイトのディスク領域が必要です。

DBA は、アウトラインが無効としてマークされていないか判断するために、格納されたすべてのアウトラインを定期的に確認してください。

5.9.9 アウトラインの削除

例 5-16（5-79）は、アウトラインの削除に SQL の DROP OUTLINE 文を使用する方法を示しています。

例 5-16 SQL の DROP OUTLINE 文を使用してアウトラインを削除する方法

```
SQL> DROP OUTLINE DEGREES_FOR_EMPS_OVER_65;
```

アウトラインを削除するには、アウトラインが参照するすべてのテーブルに対する SQL の DROP 権限を持っていることが必要です。DROP OUTLINE 文は、オンライン操作です（アウトラインを削除している間も、他のユーザーはデータベースにアタッチすることができます）。

6

VMScluster 環境で Oracle Rdb を使用する方法

OpenVMS VAX OpenVMS Alpha

VMScluster 環境では、Oracle Rdb は、マルチプロセッサによる同時データベース・アクセスを行うことができます。Oracle Rdb は、VMScluster システムのプロセッサで障害が発生すると、自動的にデータベースのリカバリを行い、オプションの After-image ジャーナルを使用して VMScluster データベースの整合性をより一層強く保護します。

正しく構成された VMScluster 環境では、Oracle Rdb はデータベースに対してほとんど不断の可用性を提供できます。

> **注意**： この章の VMScluster 環境に関する説明は、特に指定がなければ VAX ノードのみを含む VAXcluster システムと少なくとも 1 つの Alpha ノードを含む VMScluster システムの両方に適用されます。

この章では、次の内容について説明します。

- Oracle Rdb にとって重要な VMScluster の用語と概念
- VMScluster 環境で Oracle Rdb がどのように動作するか
- ローカル・エリア VMScluster 環境で Oracle Rdb がどのように動作するか
- VMScluster データベースを正しく構成する方法
- シングルノード・データベースを VMScluster データベースに変換する方法
- VMScluster データベースを監視、メンテナンスする方法

さらに、この章では次の例も示します。

- VMScluster 環境での mf_personnel データベースの作成
- mf_personnel データベースを VMScluster データベースに変換

VMScluster 環境で Oracle Rdb データベースを使用する前に、この章全体を十分に理解してください。VMScluster 環境を初めてご使用になる場合は、OpenVMS のドキュメント・セットに詳細が記載されておりますので参照してください。

VMScluster 環境での Oracle Rdb のインストール方法について疑問がある場合には、『Oracle Rdb7 Installation and Configuration Guide』を参照してください。◆

6.1 VMScluster 環境の概要

OpenVMS VAX OpenVMS Alpha

この項では、Oracle Rdb にとって重要な VMScluster の用語と概念について概要を紹介しますが、VMScluster 環境全体の説明に代わるものではありません。VMScluster 用語と概念の詳細は、OpenVMS のドキュメント・セットを参照してください。

この項には、次のトピックが含まれています。

- VMScluster 環境の一般的な定義
- 共有ディスク・ファイルのアクセス方法
- デュアルポート・ディスク
- CPU とディスク間のデュアル・パス
- デバイス・ネーミング規則
- 共通システム・ディスク
- OpenVMS ロック・マネージャ
- 分散トランザクション
- クライアント / サーバー・コンピューティング
- パーティション・データ・アクセスと共有データ・アクセス ◆

6.1.1 VMScluster 環境の定義

OpenVMS VAX OpenVMS Alpha

VMScluster システムは、ソフトウェア、VAX コンピュータと Alpha コンピュータ、およびストレージ・デバイスが高度に統合化された構成を意味しています。クラスタ化された VAX や Alpha プロセッサで動作する VMScluster ソフトウェアでは、ユーザーやアプリケーションは、高度に統合化された単一のコンピューティング環境を持つことができます。この環境では、プロセッサ、ストレージ・デバイス、バッチ・キューと印刷キュー、およびその他のリソースを可能な限り最も効率的な方法で共有することができます。

VMScluster システムでは、柔軟性の高い方法ですべてのサイズ（デスクトップからデータ・センターまで）のコンピュータを構成できます。VMScluster システムには、OpenVMS VAX か OpenVMS Alpha オペレーティング・システムが必要です。VMScluster ソフトウェアは、CI、Ethernet、DSSI、FDDI またはこれらのリンク・オプションの任意の組合せを通

してクラスタ通信を取り扱うシステム構成で動作します。OpenVMS ドキュメント・セットでは、異なるタイプの VMScluster 構成を説明しています。◆

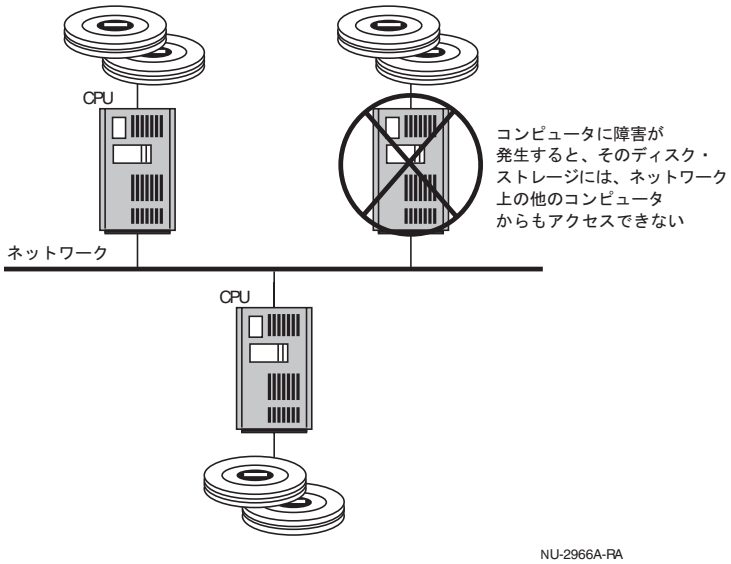
6.1.2 共有ストレージ・デバイス

OpenVMS VAX OpenVMS Alpha

VMScluster ソフトウェアの最も重要な機能は、複数のシステムにまたがって透過性の高い共有デバイスとファイルを提供する能力であり、これによってクラスタの可用性が高まります。

従来のネットワーク・システム構成では、他のシステムとネットワーク化されている場合でも、1 つのシステムにその I/O デバイスが直接接続されていました。そのため、そのシステムにアクセスできないときは、ネットワーク上の他のシステムは、そのシステムのディスク（またはそのシステムに接続されているその他のデバイス）にもアクセスできませんでした。図 6-1（6-3）に示す従来の構成では、十分な可用性は得られません。

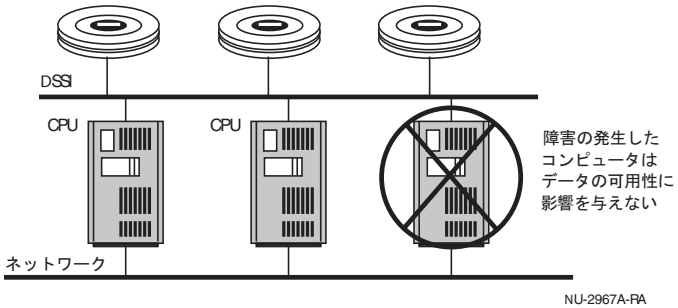
図 6-1 従来のネットワーク・システムの構成



VMScluster 構成は、I/O デバイスをシステムに直接接続するという従来のモデルを使用するのではなく、デバイスを複数のシステムからアクセスできる通信インターコネクトに接続します。あるシステムがシャットダウンしても、残りのシステムはそのデバイスに引き続きアクセスできます。VMScluster ソフトウェアは、CI、DSSI、Ethernet、FDDI の 4 つの通信インターコネクトを使用できます。ディスクやテープ・ストレージ・サブシステムによる直接通信が許可されているので、CI と DSSI は他のものとは異なります。たとえば、一般的な VMScluster 構成では、DSSI は、数台のコンピュータ・システムと数台のストレージ・サブシ

ステムで構成できます。各コンピュータ・システムは、すべてのストレージ・サブシステムへ直接アクセスできます。あるシステムでシャットダウンや障害が発生しても、他のシステムからそのストレージにアクセスする機能には、まったく影響がありません。この機能は、図 6-2（6-4）で示すように、システムとストレージの高可用性という結果をもたらします。

図 6-2 VMScluster 構成



VMScluster 環境には、次に示す 2 つのタイプのディスク・デバイスがあります。

- アクセス制限ディスク
  アクセス制限ディスクに接続されたノードまたは複数のノードだけがこれらのディスクにアクセスできます。VMScluster 環境では、OpenVMS は、ディスクがクラスタ全体に渡ってマウントされ、クラスタ・アクセス対応ディスクに設定されていない限り、各ディスク・デバイスをアクセス制限ディスクとして取り扱います。
- クラスタ・アクセス対応ディスク
  VMScluster 構成内の任意のノードは、クラスタ・アクセス対応ディスクにアクセスできます。

VMScluster 内のすべてのノードから MOUNT/CLUSTER コマンドを発行して、ディスクをクラスタ全体に渡ってマウントすると、そのディスクはクラスタ・アクセス対応ディスクになります。MOUNT コマンドの詳細情報は、OpenVMS ドキュメンテーション・セットを参照してください。◆

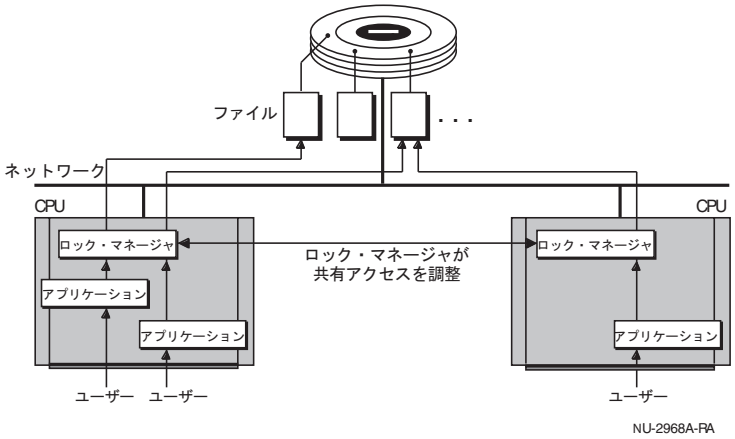
6.1.3 共有ディスク・ファイル

OpenVMS VAX OpenVMS Alpha

VMScluster ソフトウェアは、図 6-3（6-5）で示すように共有ファイルの機能も提供します。VMScluster 内のすべてのシステムは、クラスタ全体に渡ってマウントされた任意のディスク上のファイルにアクセスできます。また、VMScluster 内の複数のシステムは、完全に調整された形式で共有ファイルに同時に書き込めます。この協調作業は、OpenVMS のロック・マネージャが提供します。複数のシステムで 1 つのシステム・ディスクを共有することもできます。複数のシステムがこのディスクからブートして、オペレーティング・システ

ム・ファイルとユーティリティを共有できます。この機能を使用すると、ディスク容量を節約し、システム管理を大幅に簡易化できます。

図 6-3 ディスクとファイルの共有



◆

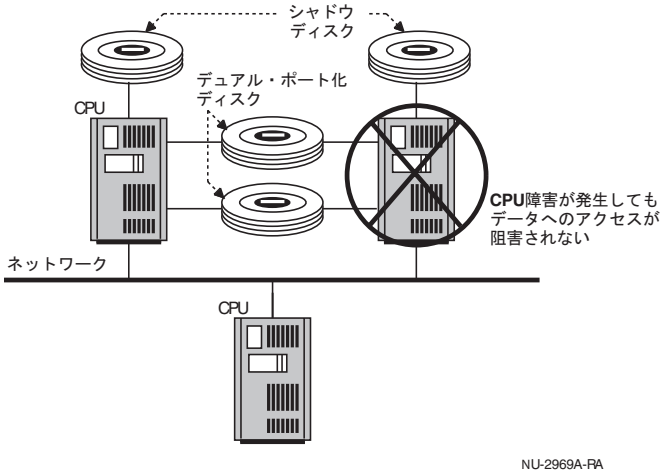
6.1.4 デュアルポート・ディスク

OpenVMS VAX OpenVMS Alpha

一般的な VMScluster システムは DSSI バスまたは CI バス上のストレージ・システムで構成されていますが、ストレージを直接特定のシステムに接続することも可能です。従来のネットワーク構成では、ホスト・システムの信頼性に頼るため、より低い可用性を提供することになります。しかし、VMScluster ソフトウェアは、ディスク・ドライブを独立した 2 つのシステムに接続するデュアルポート化を可能にすることで、これらの構成の可用性を拡張できます（図 6-4（6-6）を参照）。したがって、少なくとも 1 つのシステムが利用可能であれば、VMScluster 内のその他すべてのシステムからこのディスクにアクセスできます。ディスクは VMScluster 内にある他のディスクのシャドウ・セットを使用してシャドウすることもでき、これによって可用性が拡張されます。

デュアル・ポート化によるシステム障害からの自動リカバリは、ユーザーに対して透過的であり、オペレータの介入などは必要ありません。OpenVMS オペレーティング・システムは、使用していたどちらかのパスに対応するノードで障害が発生したら、ディスク・ファイルへのアクセス要求を 1 つのパスから別のパスに自動的に切り替えます。

図 6-4 デュアルポート・ディスク



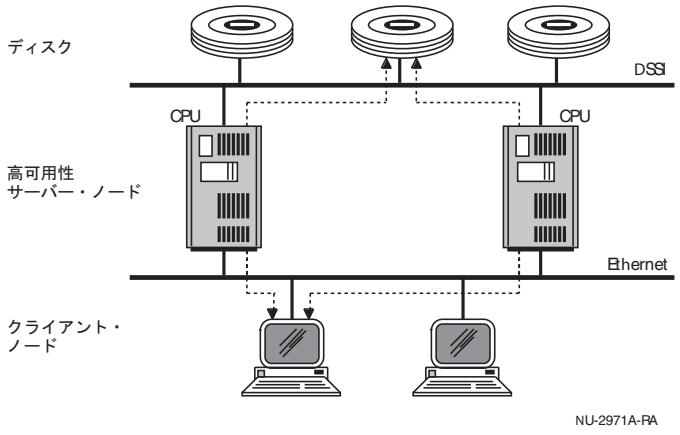
◆

6.1.5 デュアル・パス

OpenVMS VAX OpenVMS Alpha

デュアル・パスは、図 6-5（6-7）で示すように、CPU とディスクの間に複数のパスが存在することを示す VMScluster の用語です。点線は、2 つ以上のパスがあることを示しています。あるデバイスに対する 1 つのパスで障害が発生すると、OpenVMS は自動的に代替パスにフェイル・オーバーします。この機能を使用すると、高可用性システムの構築も可能です。

図 6-5 デュアルパス・ディスク



◆

6.1.6 デバイス・ネーミング規則

OpenVMS VAX OpenVMS Alpha

OpenVMS オペレーティング・システムには、3 つのデバイス・ネーミング規則があります。

- デバイス名
  オペレーティング・システムで使用されるデバイス・ネーミング規則。たとえば、次のように示します。

  ```
  DUA1:
  ```

  これは、ユーザー・ノードに接続されている RA90 ディスクを参照します。

- ノード・クラス・デバイス名
  デバイス名に追加されるノードの名前です。たとえば、次のように示します。

  ```
  SHEMP$DUA1:
  ```

  これは、ノード SHEMP に接続されている RA90 ディスクを参照します。このノード・クラス・デバイス名を使用すると、OpenVMS オペレーティング・システムは、ノード SHEMP に接続された DUA1 と別のノードに接続された DUA1 を区別できるようになります。

- 割当てクラス・デバイス名
  2 つのノードに接続されているディスクに対する共通デバイス名です。たとえば、次のように示します。

  ```
  $2$DUA1:
  ```

このデバイス名は、2 つのノードに接続されている RA90 ディスクを参照します。各ノードには、割当てクラス識別子 2 が割り当てられています。OpenVMS オペレーティング・システムは、接続されているどちらのノードを通してこのディスク上のファイルへのアクセスするかを決めるアクセス・パスを選択できます。どちらかのノードで障害が発生すると、OpenVMS オペレーティング・システムは自動的に別のノードを通るアクセス・パスに切り替えます。割当てクラスの識別子を正しく使用すると、OpenVMS オペレーティング・システムの自動フェイルオーバー機能の利点を活用できます。そして、データベースを参照するときに割当てクラス名またはノード・クラス名を使用できます。

しかし、デュアルポート・ディスクを使用している場合、（例 6-6（6-21）で示すように）データベース・ファイルの参照に割当てクラス名に変換できる論理名を使用すべきです。OpenVMS オペレーティング・システムが割当てクラス名を認識しても、ノード・クラス名を認識できないことがあります。たとえば、これは 1 つのディスクが接続されているノードの 1 つが、ブートされていない場合に起こります。◆

6.1.7 共通システム・ディスク

OpenVMS VAX OpenVMS Alpha

VMScluster 構成を使用すると、同じシステム・ディスクを複数のノードが使用できます。Alpha プロセッサと VAX プロセッサの両方が存在する VMScluster では、Alpha プロセッサは Alpha システム・ディスクからブートする必要がありますし、VAX プロセッサは VAX システム・ディスクからブートする必要があります。しかし、VAX プロセッサと Alpha プロセッサは、任意のディスクにマウントしてファイルへのアクセスを共有できます。

クラスタ内にある 2 つ以上のプロセッサが共通システム・ディスクを使用している場合、共通 SYSUAF.DAT ファイルも使用できます。共通 SYSUAF.DAT ファイルを使用すると、同じシステム・ディスクを使用して、同じユーザー名、パスワードおよびデフォルトのアカウントを使用する任意のノードにログインできるようになります。データベース・ユーザーに共通な SYSUAF.DAT ファイルを作成する場合、デフォルトのディレクティブがログインできる任意のノードからアクセス可能な共有ディスク・デバイスに存在することを確認してください。SYSUAF プロセスの数の制限と割当てのデフォルト値は、Alpha コンピュータのほうが VAX コンピュータ上での値よりも大きいことに注意してください。一般的に、共通の SYSUAF.DAT ファイルの値は、クラスタ内の最大要求に適応する必要があります。より少ない割当てでよいノードでは、ローカルな MODPARAMS.DAT ファイルを編集してシステム・パラメータをより適切な値に調整します。Alpha コンピュータや VAX コンピュータに対するプロセス割当ての決定には、OpenVMS システム管理ドキュメントを参照してください。◆

6.1.8 OpenVMS ロック・マネージャ

OpenVMS VAX OpenVMS Alpha

OpenVMS ロック・マネージャを使用すると、クラスタ全体に渡ってリソースを同期することができます。Oracle Rdb は、ロック・マネージャを使用して、クラスタ全体に渡ってデータベース・ファイルのルート部分に対する更新の同期を取り、ノードで障害が発生した場合は自動リカバリ・プロセスを起動し、異なるノードで動作するプロセスからの同じデータベースに対する同時更新を調整します。◆

6.1.9 分散トランザクション

OpenVMS VAX OpenVMS Alpha

OpenVMS オペレーティング・システムには、トランザクション処理を容易に行う DECdtm サービスと呼ばれるサービス・セットがあります。DECdtm システム・サービスを使用すると、アプリケーション設計者が複数のノードのクラスタやネットワークに渡るアトミック・トランザクションを実装できるようになります。このサービスは、2 フェーズ・コミット・プロトコルを使用します。このサポートでは、Oracle CODASYL DBMS や Oracle Rdb ソフトウェアなど、1 つのトランザクションに結合される複数のリソース・マネージャが利用できます。Oracle Rdb での DECdtm システム・サービスの使用方法の説明は、『Oracle Rdb7 Guide to Distributed Transactions』を参照してください。◆

6.1.10 クライアント / サーバー・コンピューティング

OpenVMS VAX OpenVMS Alpha

VMSclusters は、クライアント / サーバー・アプリケーション用に使用できます。アプリケーション設計者は、各ノード上で動作しており、VMScluster システムまたは広域ネットワーク上のどこかにあるノード上で実行しているクライアントからの要求を受け取るサーバー・アプリケーションを構築できます。

アプリケーションが動作しているノードで障害が発生したら、そのサーバーのクライアントは生存しているノード上で動作している別のサーバーに切り替わることができます。新しいサーバーは、障害が発生したサーバーがアクセスしていたディスクまたはテープ上の同じデータにアクセスできます。また、OpenVMS ボリューム・シャドウイング・ソフトウェアは、ディスク・コントローラやメディアで障害が発生した場合にデータが入手できなくなるのを防止します。VMScluster は、クライアント・アプリケーションにとって非常に高い可用性があります。◆

6.1.11 パーティション・データ・アクセスと共有データ・アクセス

OpenVMS VAX OpenVMS Alpha

VMScluster では、ディスク・ストレージ・デバイスはすべてのノードからアクセスできます。アプリケーション設計者は、アクセスが一度に 1 つのノードから（パーティション・データ・アクセス）か複数のノードから（共有データ・アクセス）同時に起こるかを選択できます。

パーティション・データ・モデルを使用する場合、アプリケーション設計者はデータ・アクセスを 1 つのノードまたはノードのサブセットに制限するアプリケーションを構築できます。アプリケーションは、1 つのノード上のサーバーとして実行し、ネットワーク内の他の

ノードからの要求を受け取ります。アプリケーションは 1 つのノード上で動作しているので、他のノードとのデータ・アクセスの同期を取る必要はありません。ノード間でのデータ・アクセスの同期に関連するオーバーヘッドを避けると、多くのアプリケーションのパフォーマンスを改良することができます。同期を取る必要がなければ、アプリケーション設計者は、ノード上のバッファ・キャッシュを最高に利用して、書込み操作用の大量のデータを収集できるようにすることで、I/O アクティビティを最小化します。

パーティション・データ・アクセスを使用するアプリケーションは、多くのタイプの高パフォーマンス・データベースとトランザクション処理環境に貢献しています。VMScluster システムは、そのようなアプリケーションに対し、データ・ファイルにアクセスしているときでもすべてのノードで利用できるストレージ・メディアを持つ利点を提供します。したがって、サーバーのノードで障害が発生したら、生存しているノード上の別のサーバーが作業を引き継ぎ、同じファイルにアクセスできます。このタイプのアプリケーション設計の場合、単一サーバーでの障害に関連する問題に関係なく、VMSclusters はパーティション・データ・モデルの利点であるパフォーマンスが提供されます。

共有データ・モデルを使用する場合、アプリケーション設計者は 1 つのファイル内のデータを自然に共有する VMScluster 内の複数のノードで同時に動作するアプリケーションを作成できます。このタイプのアプリケーションは、1 つのサーバーに関連したボトルネックを避けることができ、複数のプロセッサによる並列機能を利用します。OpenVMS RMS ソフトウェアは、VMScluster 内の複数のノード間で透過的にファイルを共有できます。Oracle Rdb と Oracle CODASYL DBMS には、同じデータ共有機能があります。VMScluster システムの複数のノード上で動作しているサーバーは、ネットワーク内のクライアントからの要求を受け入れ、同じファイルやデータベースへアクセスできます。サーバーは複数台あるので、1 つのサーバー・ノードで障害が発生しても、アプリケーションは機能します。◆

6.2 VMScluster 環境内の Oracle Rdb

OpenVMS VAX OpenVMS Alpha

VMScluster 環境に Oracle Rdb をインストールする方法の詳細は、『Oracle Rdb7 Installation and Configuration Guide』を参照してください。Oracle Rdb のバージョン 4.1 以降、VMScluster 内の 1 つのノードまたは 1 つ以上のノードで複数のバージョンの Oracle Rdb をインストールして実行できるようになりました。特定のバージョンの Oracle Rdb を使用する各ノードで、そのノード上の Oracle Rdb のバージョンで共有できるイメージをインストールする必要があります。

この項では、VMScluster 環境内の Oracle Rdb の操作とシングルノード環境内の Oracle Rdb の操作を比較します。◆

6.2.1 シングルノード環境と VMScluster 環境

OpenVMS VAX OpenVMS Alpha

同じデータベースに 2 つ以上のノードから同時にアクセスする場合、VMScluster 環境で Oracle Rdb を使用します。この環境では、Oracle Rdb は、各ノードで動作しているモニタープロセス間で 2 つ（またはそれ以上）のノード間に分散ルート・ファイル・アクセスを確立しており、OpenVMS ロック・マネージャを通した通信を確立しています。これは、各ノードのモニター・プロセスが他のノードのデータベース・ユーザーを認識していることを保証しています。

Oracle Rdb の目的によると、データベースは次のように定義できます。

- データベース・ユーザーのすべてがシングルノード上に存在すれば、シングルノード環境にあります。
- データベース・ユーザーが同時に 2 つ以上のノードに存在すれば、VMScluster 環境にあります。◆

6.2.2 VMScluster 環境でデータベースへのアクセスと利用を可能にする

OpenVMS VAX OpenVMS Alpha

VMScluster 環境で Oracle Rdb を使用する場合、できる限りデータベースへのアクセスと利用を可能にしようと考えます。次のステップに従うと、データベースにアクセスできるようになります。

1. すべてのデータベース・ファイル（.rdb、.rda、.snp、.ruj および .aij）をクラスタ・アクセス対応のディスク・ドライブ（ディスクは MOUNT/CLUSTER コマンドでマウント）に置きます。ディスクがどのノードに接続されているかにかかわらず、クラスタ内の任意のノードからクラスタ・アクセス対応ディスクにアクセスできます。Oracle Rdb データベース・ファイルが設置される場所についてのガイドラインは、6.2.6 項（6-16）を参照してください。
2. データベースへのアクセスに必要な VMScluster の各ノードに適切なバージョンの Oracle Rdb が正しくインストールされているかを確認してください。VMScluster ノード上に Oracle Rdb をインストールし、起動するための情報は、『Oracle Rdb7 Installation and Configuration Guide』を参照してください。
3. SQL の CREATE DATABASE、ALTER DATABASE または IMPORT 文の NUMBER OF CLUSTER NODES オプションに適切な値を指定します。データベースに NUMBER OF CLUSTER NODES 値を指定するための情報は、6.2.3 項（6-12）を参照してください。
4. Oracle CDD/Repository を使用可能にします（データベースで Oracle CDD/Repository を使用する予定であれば）。6.2.7 項（6-19）を参照してください。
5. Oracle Rdb をローカル・エリア VMScluster 構成で使用する計画であれば、6.6.1 項（6-30）のローカル・エリア VMScluster 構成に対する特別考慮点を参照してください。

次のステップに従うと、データベースを利用できるようになります。

1. いつでもデータベース・ファイル（.rdb、.rda、.snp、.ruj および .aij ファイル）が使用できるようにデュアルポート HSC ディスクを使用します。Oracle Rdb データベース・ファイル用のデバイス構成の推奨は、6.2.6 項（6-16）を参照してください。
2. .aij ファイルは他のデータベース・ファイルとは異なるディスクに置いてください。たとえば、.rdb、.rda、.snp または .ruj ファイルが含まれているディスク上に .aij ファイルを置かないでください。詳細は、8.1.2.2 項（8-6）を参照してください。◆

6.2.3 VMScluster ノードを最大数に指定

OpenVMS VAX OpenVMS Alpha

SQL の CREATE DATABASE、ALTER DATABASE および IMPORT 文に NUMBER OF CLUSTER NODES オプションを使用します。

- .rdb ファイルのサイズを制御します（各 VMScluster ノードが追加されると、1 ブロック増えます。この項の後で示す例を参照）。
- ユーザーがデータベースにアクセスするために使用する VMScluster ノードの数の上限を設定してください。

NUMBER OF CLUSTER NODES オプションの範囲は、1 から nVMScluster ノードまでです。ここで、n は現行の OpenVMS ソフトウェアの制限です。OpenVMS 制限よりも高い値を選択すると、その値はデータベース・ファイルの最初のサイズの計算に使用され、Performance Monitor の「General Information」画面上に表示されます。さらに、実際の制限は、VMScluster 構成のタイプを基準にした現行の VMScluster ソフトウェアによって許容されているノードの最大数です。

NUMBER OF CLUSTER NODES IS 17 オプションで作成されたデータベースは、NUMBER OF CLUSTER NODES IS 10 オプションで作成された同じデータベースよりも大きくなります。Oracle Rdb ユーザーが共有 Oracle Rdb データベースにアクセスする VMScluster 環境の各ノードでは、.rdb ファイル内にデータ構造を追加する必要があります。

SQL の CREATE DATABASE、ALTER DATABASE および IMPORT 文から NUMBER OF CLUSTER NODES オプションを省略すると、デフォルトは 16 になります。IMPORT 文でこのオプションを使用すると、以前のバージョンの Oracle Rdb で作成したデータベースに新しい値を設定できます。

SQL の CREATE DATABASE 文と ALTER DATABASE 文で NUMBER OF CLUSTER NODES オプションを使用する場合、続く SQL の EXPORT 操作と IMPORT 操作は、パラメータの値を .rbr ファイル内に保存するので SQL の EXPORT 文に続いて .rbr ファイル内にこの値を保持します。

VMScluster ノードからデータベースにアクセスしようとして、実行に移したときに最大ノード・パラメータを超えると、例 6-1（6-13）に示すエラーが発生します。

例 6-1 最大ノード・パラメータの超過

```
SQL> -- VMScluster max nodes value is 5,
SQL> -- and database is already at the limit:
SQL> ATTACH 'FILENAME $222$DUA12:[TOP]mf_personnel';
%RDB-F-SYS_REQUEST, error from system services request
-RDMS-F-NORTUPB, no more user slots are available in the database
SQL>
```

例 6-2（6-13）では、単一ファイルの personnel データベースを作成し、NUMBER OF CLUSTER NODES IS オプションを使用して VMScluster ノードの数に値 20 を定義します。20 ノードのローカル・エリア VMScluster 構成の共有ディスクにデータベースが存在する場合には、このようにデフォルトの 16 から 20 の VMScluster ノードへ変更できます。例 6-3（6-13）は、ノード・カウントの新しい値を表示する Performance Monitor の「General Information」画面を示しています。

例 6-2 NUMBER OF CLUSTER NODES 値の指定

```
SQL> CREATE DATABASE FILENAME personnel NUMBER OF CLUSTER NODES IS 20;
SQL> EXIT
```

例 6-3 NUMBER OF CLUSTER NODES 値の最大値を表示

```
Node: TRIXIE          Oracle Rdb V7.0-00 Performance Monitor 24-JUN-1996 14:15:32
Rate: 3.00 Seconds            General Information           Elapsed: 00:00:16.86
Page: 1 of 1         RDBVMS_USER1:[LOGAN.V70]PERSONNEL.RDB;1         Mode: Online
------------------------------------------------------------------------------
Database created at 24-JUN-1996 14:14:46.13
Maximum user count is 50
Maximum node count is 20       <------------ 注意
Database open mode is Automatic
Database close mode is Automatic
Snapshot mode is Automatic
Statistics collection is enabled
Default recovery-unit journal filename is "Not Specified"
Date of last backup is 17-NOV-1858 00:00:00.00
Fast incremental backup is enabled
------------------------------------------------------------------------------
Exit Help Menu Options Refresh Set_rate Write !
```

例 6-4（6-14）は、SQL の IMPORT 文で NUMBER OF CLUSTER NODES オプションを使用して、ノード・カウント・パラメータの値を 15 に減らします。マルチファイル・データベースの場合、SQL の ALTER DATABASE 文を使用して、NUMBER OF CLUSTER NODES オプションに新しい値を指定できます。

例 6-4 NUMBER OF CLUSTER NODES 値の変更

```
$ SQL
SQL> EXPORT DATABASE FILENAME personnel.rdb INTO personnel_test.rbr
  1> WITH EXTENSIONS;
SQL> IMPORT DATABASE FROM personnel_test.rbr
  1> FILENAME personnel_test.rdb
  2> NUMBER OF CLUSTER NODES 15;     <---- 15 に削減
Exported by Oracle Rdb V7.0-00 Import/Export utility
A component of SQL V7.0-00
Previous name was personnel.rdb
It was logically exported on 28-MAY-1996 15:29
   .
   .
   .
Database NUMBER OF CLUSTER NODES was 20, now is 15
   .
   .
   .
```

◆

6.2.4 複数のモニター・プロセス

OpenVMS VAX OpenVMS Alpha

シングルノード環境では、Oracle Rdb は分離モニター・プロセスを使用してデータベース・リカバリの制御、データベースのオープンとクローズ操作の調整およびアクティブ・データベース・ユーザーのリストの維持を行います。

VMScluster 構成では、プロセスのノード間での共有を許容していないので、Oracle Rdb は、Oracle Rdb データベースにアクセスできる VMScluster システム内の各プロセッサ上でデータベース・モニター・プロセスを使用します。このモニター・プロセスは、OpenVMS ロック・マネージャを通して互いに通信します。VMScluster 環境内の 1 つのノード上のモニター・プロセスは、他のノード上のデータベース・ユーザーを識別します。

Oracle Rdb データベースにアクセスする各プロセッサのシステム起動ファイルに @SYS$STARTUP:RMONSTART 行を加える必要があります。Oracle Rdb の各バージョンごとにモニター・プロセスがあるため、複数バージョン環境では、インストールされている各バージョンの Oracle Rdb に対して別々の RMONSTART.COM ファイルがあります。したがって、複数バージョン環境では、バージョンに適切な RMONSTART.COM ファイルを指定することで、特定のバージョンの Oracle Rdb に対する監視を開始します（たとえば、Oracle Rdb バージョン 6.1 の監視を開始するには、@SYS$STARTUP:RMONSTART61 を指定します）。RMONSTART.COM を使用して Oracle Rdb を起動するための説明は、『Oracle Rdb7 Installation and Configuration Guide』を参照してください。

VMScluster 環境でプロセッサに障害が発生した場合、OpenVMS ロック・マネージャは同じデータベースのユーザーが存在する他のノード上の Oracle Rdb モニター・プロセスに警告を送ります。これらのモニター・プロセスの 1 つが、Oracle Rdb リカバリ・プロシージャを

起動し、障害が発生したノード上のユーザーが進行中であったトランザクションをロールバックします。Oracle Rdb データベース・リカバリ・プロシージャの詳細は、6.5 項（6-27）を参照してください。◆

6.2.5 パーティション・ロック・ツリー

OpenVMS Alpha

デフォルトでは、Oracle Rdb データベース内のすべてのリソースに対するアクティブ・ロックは、そのデータベースに対するデータベース・ロックの下にあるシングル・リソース・ツリーに属します。VMScluster 環境では、データベースのリソース・ツリー内のすべてのロックとリソースは、VMScluster 内のノードの 1 つの OpenVMS が所有します。

OpenVMS は、すべてのロックとリソースを所有する VMScluster ノードを選択します。この選択は、種々のロックの使用によって影響を受ける可能性があります。1 つのノードのみがデータベース・リソース・ツリーを所有できるので、VMScluster 内の他のノードは各ロック要求に対して、マスター・ノードとメッセージを交換する必要があります。1 つのノードが VMScluster 環境内のロック要求に関連したすべてのメッセージを高いトランザクション・レートで処理する場合、パフォーマンスのボトルネックが発生する可能性があります。

データベースが Oracle Rdb for OpenVMS Alpha データベースであれば、例 6-5（6-15）で示すように、SQL の CREATE 文または ALTER DATABASE 文で LOCK PARTITIONING IS ENABLED 句を使用して、データベースごとのシングル・リソース・ツリーによって発生するパフォーマンスのボトルネックを緩和できます。

例 6-5 ロック・パーティション化を有効にするには

```
SQL> CREATE DATABASE FILENAME pers_test
  1>       LOCK PARTITIONING IS ENABLED;
```

データベースに対するパーティション・ロック・ツリーを有効にする場合、データベース内のすべてのリソースを含む 1 つのリソース・ツリーがデータベースごとに存在するわけではありません。領域ロックは、データベース・リソース・ツリーとは分離されており、リソースに対するトラフィックが最も高い VMScluster ノード上に独立して所有されています。OpenVMS は、各リソースを最もよく使用するノードを決定して、リソース階層（領域ロックと基盤のページ・ロックおよびレコード・ロック）をそのノードに移動します。

パーティション・ロック・ツリーを有効にして得られたパフォーマンスの向上は、データベースを使用するパーティション・アプリケーションにとっては特に重要です。たとえば、VMScluster ノードの NODE1 が記憶領域 AREA1 にアクセスする唯一のノードであるパーティション・アプリケーションが存在する場合、NODE1 は AREA1 に対する他の VMScluster ノードからのロック要求を取り扱う必要はありません。NODE1 は他のノードからの AREA1 に対するロック要求を取り扱う必要がないので、パーティション・ロック・ツリーが有効な場合には、このアプリケーションはさらに高速で動作します。

ALTER 文または CREATE DATABASE 文で LOCK PARTITIONING IS DISABLED 句を使用すると、パーティション・ロック・ツリーを無効にできます。

SQL 構文の詳細は、『Oracle Rdb7 SQL Reference Manual』の CREATE DATABASE 文と ALTER DATABASE 文を参照してください。◆

6.2.6 VMScluster 環境で Oracle Rdb ファイルの設置場所を決定するには

OpenVMS VAX OpenVMS Alpha

この項では、VMScluster 環境で Oracle Rdb ファイルを設置する場所を決定する際の推奨方法を示します。これらの推奨の目的は次のとおりです。

- VMScluster 環境の適切なノードからデータベース・ファイルへのアクセスを提供
- ノード障害が発生した場合の自動データベース・リカバリ
- データベースに対して実質的に中断のない可用性を保証

Oracle Rdb リカバリ・プロセスが、VMScluster 環境で正しく動作するためには、すべての Oracle Rdb データベースとジャーナル・ファイルが、データベースを使用するクラスタ内のすべてのノードから常にアクセス可能なディスク上に存在する必要があります。クラスタからアクセスできるデバイス上にすべてのデータベース・ファイルとジャーナル・ファイルが置かれていなければ、Oracle Rdb はノードで障害が発生したときにデータベース・リカバリ・プロシージャを起動して完了することができない可能性があります。Oracle Rdb がデータベースをリカバリできない場合、データベースに対するすべてのアクセスはリカバリが完了するまで停止します。

Oracle Rdb がデータベースをリカバリした後、障害が発生したノード上のユーザーは他のノードにログインし、引き続きデータベースを使用できます。したがって、ユーザーが生存ノードに対するアカウントを持っているか、または VMScluster が共通 SYSUAF.DAT ファイルを使用していることを確認する必要があります。トランザクション・ロールバックの詳細は、6.5 項（6-27）を参照してください。

次に示す VMSclusters の様々なオプションから、高パフォーマンス・アクセスと高可用性を提供するオプションを選択できます。

- デュアルポート HSC ディスク
  HSC サブシステムの 1 つで障害が発生したら、ファイルに対する高パフォーマンスおよび可用性の両方を持つ代替パスを提供します。
- シングルポート HSC ディスク
  高パフォーマンスではありますが、ファイルに対する代替パスは自動的には提供されません。
- デュアルポート接続ディスク
  接続ディスクは、ローカルに接続されたディスクであり、MSCP サーバーを使用してクラスタ・アクセス対応として明示的に設定されています。接続ディスクの詳細は、OpenVMS システム管理ドキュメントを参照してください。
  デュアルポート接続ディスクは、VAX プロセッサまたは Alpha プロセッサの 1 つで障害が発生した場合に、自動的にファイルに対する代替パスを提供します。しかし、接続ディスクは、接続されているプロセッサに対してのみ高パフォーマンスを提供します。

- シングルポート接続ディスク
  これを使用すると、VMScluster 環境内のすべてのプロセッサで Oracle Rdb ファイルが利用可能になります。しかし、接続されているプロセッサに対してのみ高パフォーマンスを提供しますが、ファイルに対する代替パスは提供しません。

VMScluster 構成内のノードで障害が発生したときに、生存ノードを使用してデータベース・アクセスの継続を保証するため、Oracle Rdb ファイルを、シングルポート接続ディスクにクラスタマウントされたディスク・デバイス上に配置しないでください。シングルポート・ディスク上に Oracle Rdb ファイルを配置して、そのディスクが接続されているノードで障害が発生した場合、Oracle Rdb リカバリ・プロシージャはこれらのファイルにアクセスできません。そのデータベースへのアクティビティは、障害が発生したノードが再開するまで中断します。

ノードで障害が発生したことにより、データベースへのアクセスが失われないようにするには、次の事項を考慮してください。

- ファイルをデュアルパス HSC ディスクに配置すると、データベースへアクセスできなくなるのは両方の HSC サブシステムで同時に障害が発生した場合のみです。
- ファイルをデュアルパス接続ディスクに配置すると、データベースへアクセスできなくなるのは両方のプロセッサで同時に障害が発生した場合のみです。
- ファイルをシングルパス HSC ディスクに配置すると、データベースへアクセスできなくなるのは HSC サブシステムで障害が発生した場合のみです。
- ファイルを HSC サブシステム以外のシングルパス・ディスクに配置すると、データベースへアクセスできなくなるのは VAX または Alpha プロセッサで障害が発生した場合のみです。

デュアルパス・ディスクを使用している場合、データベース・ファイルを参照するときは常に割当てクラス名の論理名を使用します。ディスクの割当てクラス名の説明は、6.1.6 項（6-7）を、割当てクラス名に対する論理名の使用方法の説明は、例 6-6（6-21）を参照してください。

注意：ローカル・エリア VMScluster 構成の場合、ブート・ノードの 1 つで障害が発生すると、そのノードがクラスタに再結合するまで、クラスタ操作は停止されます。この状態は通常のものであり、共有クラスタ・リソースの整合性を保証します。

次のガイドラインは、データベース・ファイルを配置する場所の決定を支援します。

- データベース・ルート（.rdb）、記憶領域（.rda）およびスナップショット（.snp）・ファイル
  シングルノード環境では、Oracle Rdb はデータベース・ファイルのルート部分、またはルート・ヘッダーをメモリーのグローバル・セクションとしてマップします。

VMScluster 構成は、ノード間におけるメモリーの共有を許していないため、VMScluster 環境の Oracle Rdb はページファイル・セクション内にあるルート・ヘッダーのコピーをデータベースを使用する各ノード上で維持しています。
ルート・ファイル自体は、作成したディスク上に残ります。したがって、このディスクはデータベースにアクセスする VMScluster システム内のすべてのノードからアクセスできる必要があります。データベースのルート・ファイルが、あるノードで利用できなければ、そのデータベースはこのノードでは使用できません。
OpenVMS ロック・マネージャは、すべてのルート・ファイルのコピーが一意であることを保証します。
ルート、記憶領域およびスナップショット・ファイルは、データベースにアクセスするすべてのノードからアクセスできる必要があります。Oracle Rdb は、VMScluster 分散ルート・ファイル・アクセスを作成し維持できる必要があります。

- Recovery-unit ジャーナル（.ruj）・ファイル
  Oracle Rdb は、ノードに障害が発生したときにその自動リカバリ手順を完了できる必要があります。リカバリは、すべての .ruj ファイルに対して、データベースにアクセスするすべてのノードからアクセスできる場合のみ完了できます。すべての .ruj ファイルがアクセスできることを保証するため、次の規則に従ってください。
  - 共通ディレクトリを参照するため、システム・テーブルの RDMS$RUJ 論理名を定義します。
  - 共通ディレクトリを参照するため、各プロセスの RDMS$RUJ 論理名を定義します。
  - .ruj ファイルをユーザーのデフォルト・ディレクトリに配置する場合、ディレクトリのデバイスはクラスタ・アクセス対応ディスクに配置する必要があります。
- After-image ジャーナル（.aij）ファイル
  .aij ファイルはデータベースにアクセスする各ノードからアクセスできる必要があります。これは、データベースにアクセスするすべてのプロセスが .aij ファイルに書き込むことができることを保証します。
  マルチユーザー・データベース・アクセスの I/O 競合の削減については、ディスク I/O 操作とデータベース・パフォーマンスに対応する効果を説明する 3.2 項（3-8）と 8.1.1 項（8-2）を参照してください。.aij ファイルをデータベース・ファイルとは別のデバイスに配置することは、データベース・ディスクの I/O 競合を最小化する 1 つの方法です。

この項で説明したすべての推奨事項は、データベースが VMScluster 内のいくつかのノードまたはすべてのノードからアクセスできるようにすること、高可用性を持つデータベースになること、およびノードで障害が発生した場合 Oracle Rdb が自動的にデータベースをリカバリする、という仮定に基づいていることに注意してください。データベースによっては、高可用性はセキュリティほど重要ではないかもしれません。たとえば、データベースがプライベート・データベースまたはパブリック・システム上の機密データベースの場合には、VMScluster の他のノードからデータベースを利用できるようにすることよりも、データベース・セキュリティに対してより関心を持つかもしれません。この場合、そのデータベースを、クラスタ全体に渡ってマウントしていないローカル・ノード上のディスクに配置する

ことができます。これで、他の VMScluster ノードのユーザーがこれらのファイルにアクセスするのを防止できます。データベース・ファイルがローカル・ノードに存在すれば、データベース要求がネットワーク上を渡ってデータベースにアクセスする必要がない上、ロックはローカル・ノードが所有しているので、パフォーマンスも向上します。しかし、データベースの可用性は犠牲になります。つまり、データベース・ファイルを含むノードまたはディスクの 1 つで障害が発生したら、ローカル・ノードに対する問題が解決するまでデータベースにアクセスできなくなります。ローカル・ノードで障害が発生すると、Oracle Rdb はノードが再起動したときにデータベースを自動的にリカバリします。◆

6.2.7 Oracle CDD/Repository 要件

OpenVMS VAX OpenVMS Alpha

データ・リポジトリである Oracle CDD/Repository がインストールされている場合に、リポジトリ定義を、必要とするすべてのノード上のすべてのプロセスで利用できなければなりません。次に、リポジトリにアクセスするプロセスのタイプを示します。

- リポジトリに影響を与えるデータ定義文を実行するプロシージャ、コマンド・ファイルおよび対話型ユーザー操作
- データベース定義のリポジトリが必要なプログラム

VMScluster 環境で Oracle Rdb データベースを作成する前に、主要リポジトリ・ファイルが、すべてのユーザーからアクセスできるデバイス上に存在することを確認する必要があります。

一般的に、主要リポジトリと互換性リポジトリは、VMScluster 構成内の任意のユーザーからアクセスできる共有可能なクラスタ・ディスクに配置します。これらのファイルを共有可能なクラスタ・ディスクに配置し、1 つの論理リポジトリを作成することで、次の利点が得られます。

- あるシステムで障害が発生した場合でも、組織内のユーザーはリポジトリにアクセスできます。
- クラスタ全体に渡って、組織がデータ定義を共有することができます。シングル・リポジトリによって、冗長なデータ・ストレージが制限され、すべてのデータベース定義に渡って一貫性を保証できるようになります。異なるシステム上に別のリポジトリを維持する場合、データの更新におけるデータ定義の不一致と非一貫性の矛盾をかかえることになります。

次の 3 つのコマンドを使用すると、リポジトリを配置する予定である共有ディスクの論理名を定義し、1 つの論理リポジトリを確立し、リポジトリ・ファイルを共有 VMScluster ディスクに配置することができます。

```
$ DEFINE/SYSTEM CDD_DISK $2$DUA11:
$ DEFINE/SYSTEM/EXECUTIVE_MODE CDD$COMPATIBILITY CDD_DISK:[CDD]
$ DEFINE/SYSTEM CDD$DICTIONARY CDD_DISK:[DMU]
```

これらのコマンドは、主要ディクショナリを 2DUA11 という名前の HSC ディスク上に配置します。

これらの論理名は、Oracle CDD/Repository リポジトリにアクセスが必要な他の名称と同様に、CDDSTRTUP.COM プロシージャで定義されます。このプロシージャは、これらの主要リポジトリ・ファイルを使用してノードを開始する各 SYSTARTUP_V5.COM または SYSTARTUP_VMS.COM によって実行されます。

主要リポジトリ・ファイルを、アクセス制限ディスク上に配置することができます。しかし、データ定義が必要なすべてのプロセスが、その場所でファイルにアクセスできることを保証する必要があります。

6.3 VMScluster 環境での mf_personnel データベースの作成

OpenVMS VAX OpenVMS Alpha

この項では、VMScluster 環境に mf_personnel データベースを分散して、他のノードからアクセスできるようにする方法を、例を示して説明します。この例で必要なデータベース・ファイルは、次のとおりです。

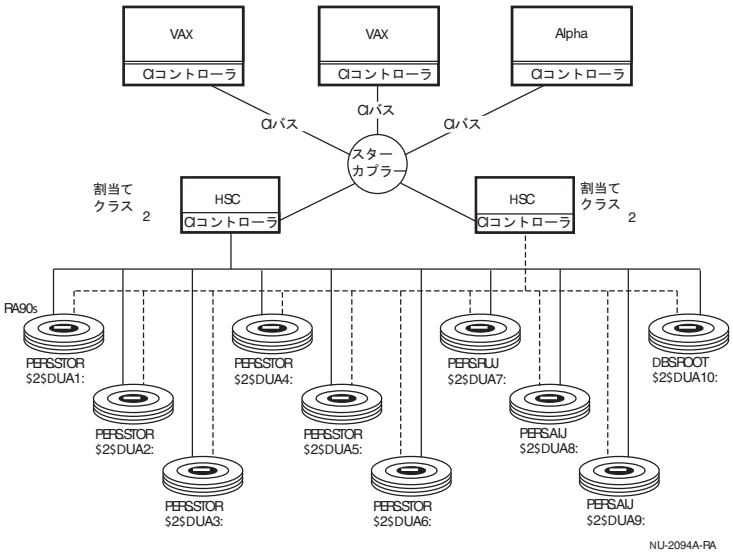
cdd.dic	mf_personnel.rdb	empids_low.rda
empids_mid.rda	empids_over.rda	salary_history.rda
departments.rda	emp_info.rda	jobs.rda
resume_lists.rda	resumes.rda	empids_low.snp
empids_mid.snp	empids_over.snp	salary_history.snp
departments.snp	emp_info.snp	jobs.snp
resume_lists.snp	resumes.snp	mf_personnel.aij (任意指定)

構成には、次のハードウェアが含まれています。

- Ten RA90 ディスク、デュアルパスによる 2 つの HSC サブシステム
- VAX プロセッサ x 3
- 拡張 VMScluster ハードウェア要件として、CI バス、CI コントローラおよびスター・カプラー

SQL の CREATE DATABASE 文とその修飾子は、これらのファイルを割当てクラス名が割り当てられた HSC ディスク上に配置します。図 6-6（6-21）は、VMScluster 環境のデータベース・ファイルを含むディスクを示しています。

図 6-6 データベース・ファイルの配置サンプル



VMScluster マネージャは、数値 2 を各 HSC サブシステムに割当てクラスの識別子として割り当てます。10 個の RA90 ディスクに対する割当てクラス名は、2DUA1、2DUA2、2DUA3、2DUA4、2DUA5、2DUA6、2DUA7、2DUA8、2DUA9 および 2DUA10 です。

例 6-6（6-21）でディスクに定義されている論理名は、図 6-6（6-21）を参照しています。

例 6-6 データベース・ファイルに対して使用しているディスクの論理名の定義

```
$ DEFINE/SYSTEM DB_DISK $2$DUA10:
$ DEFINE/SYSTEM DISK1 $2$DUA1:
$ DEFINE/SYSTEM DISK2 $2$DUA2:
$ DEFINE/SYSTEM DISK3 $2$DUA3:
$ DEFINE/SYSTEM DISK4 $2$DUA4:
$ DEFINE/SYSTEM DISK5 $2$DUA5:
$ DEFINE/SYSTEM DISK6 $2$DUA6:
$ DEFINE/SYSTEM RUJ_DISK $2$DUA7:
$ DEFINE/SYSTEM AIJ_DISK1 $2$DUA8:
$ DEFINE/SYSTEM AIJ_DISK2 $2$DUA9:
```

例 6-7（6-22）では、データベース・ファイルに対するディレクトリが定義されています。

例 6-7 データベース・ファイル用にディレクトリを作成

```
$ CREATE/DIRECTORY DB_DISK:[DBS.ROOT]
$ CREATE/DIRECTORY DISK1:[PERS.STOR]
$ CREATE/DIRECTORY DISK2:[PERS.STOR]
$ CREATE/DIRECTORY DISK3:[PERS.STOR]
$ CREATE/DIRECTORY DISK4:[PERS.STOR]
$ CREATE/DIRECTORY DISK5:[PERS.STOR]
$ CREATE/DIRECTORY DISK6:[PERS.STOR]
$ CREATE/DIRECTORY RUJ_DISK:[PERS.RUJ]
$ CREATE/DIRECTORY AIJ_DISK1:[PERS.AIJ]
$ CREATE/DIRECTORY AIJ_DISK2:[PERS.AIJ]
```

すべての .ruj ファイルを含むクラスタ・アクセス対応デバイスを参照する各プロセッサに RDMS$RUJ 論理名を定義できます。たとえば、例 6-8（6-22）内のコマンドは、そのデータベースにアクセスするすべてのプロセス用の .ruj ファイルを含む RUJ_DISK:[PERS.RUJ] ディレクトリを参照する RDMS$RUJ 論理名を定義します。

例 6-8 .ruj ファイルのディレクトリに対する RDMS$RUJ 論理名の定義

```
$ DEFINE/SYSTEM RDMS$RUJ RUJ_DISK:[PERS.RUJ]
```

.ruj ファイルに対してシステム全体のディレクトリを使用するつもりであれば、このコマンドを SYSTARTUP_V5.COM または SYSTARTUP_VMS.COM ファイルに加えてください。データベース・ファイルとは別のデバイス上にこのディレクトリを定義します。デフォルトである SYS$LOGIN を使用して、各ユーザーのデフォルト・ディレクトリ内のプロセスごとに .ruj ファイルを配置することもできます。共通ディレクトリの参照に論理名 RDMS$RUJ を定義しない場合には、クラスタ・アクセス対応ディスク上にデフォルト・ディレクトリが存在するアカウントに、すべてのデータベース・ユーザーがログインしていることを確認してください。

例 6-9（6-22）は、定義した論理名を使用する SQL の CREATE DATABASE 文の使用方法を示します。

例 6-9 サンプル・データベースの定義

```
SQL> CREATE DATABASE FILENAME 'DB_DISK:[DBS.ROOT]mf_personnel'
  1> PATHNAME 'SYS$COMMON[HEADQUARTERS]mf_personnel'
  2> UMBER OF USERS IS        50
  3> NUMBER OF BUFFERS IS      20
  4> NUMBER OF CLUSTER NODES IS 16
  5> NUMBER OF RECOVERY BUFFERS IS 20
  6> BUFFER SIZE IS             6 BLOCKS
  7> SNAPSHOT IS ENABLED
  8> DICTIONARY IS REQUIRED;
  9> DEFINE STORAGE AREA DEPARTMENTS
 10>   FILENAME DISK3:[PERS.STOR]departments.rda
 11> ALLOCATION IS 25 PAGES
```

```
 12> PAGE FORMAT IS MIXED
 13> SNAPSHOT FILENAME DISK4:[PERS.STOR]departments.snp
 14> SNAPSHOT ALLOCATION IS 10 PAGES
 15> END DEPARTMENTS STORAGE AREA;
SQL>
```

例 6-9（6-22）の文は、デュアルパス HSC ディスク・デバイス上に mf_personnel.rdb、departments.rda および departments.snp データベース・ファイルとディレクトリ名 DB_DISK:[DBS.ROOT]、DISK3:[PERS.STOR] および DISK4:[PERS.STOR] のディレクトリを作成します。mf_personnel データベースは、クラスタ内の任意のノードからアクセスできます。

SQL の ALTER DATABASE 文は、例 6-10（6-23）に示すように .aij ファイルとその位置を指定します。

例 6-10 .aij ファイルの位置を指定

```
SQL> ALTER DATABASE 'DB_DISK:[DBS.ROOT]mf_personnel'
  1> ADD JOURNAL AIJ_ONE
  2> FILENAME 'AIJ_DISK1:[PERS.AIJ]aij1.aij'
  3> ADD JOURNAL AIJ_TWO
  4> FILENAME 'AIJ_DISK2:[PERS.AIJ]aij2.aij';
SQL>
```

VMScluster システムのノードで障害が発生した場合には、生存ノード上のモニター・プロセスがデータベースのリカバリに必要な .ruj ファイルと .aij ファイルにアクセスできます。

この VMScluster 構成は、共通 SYSUAF.DAT ファイルを持つ共通システム・ディスクを使用しているので、ノードの 1 つで障害が発生しても、ユーザーは生存ノードの 1 つからログインして、データベースを継続して使用できます。

HSC サブシステムの 1 つで障害が発生したら、データベース・ファイルへのアクセスは自動的に他の HSC を通るパスに切り替わります。

このステップのリストを使用して、VMScluster 環境に Oracle Rdb データベースを作成する手順を説明します。

1. データベースへのアクセスに必要なノードを決定します。次の項目を確認する必要があります。
   a. Oracle Rdb は各ノードで起動できる
   b. Oracle Rdb 共有イメージが各ノードにインストールされている
   c. @SYS$STARTUP:RMONSTART.COM プロシージャは、各ノードが使用する SYSTARTUP_V5.COM ファイルまたは SYSTARTUP_VMS.COM ファイルに存在する
   d. 論理名 RDMS$RUJ は、データベース・アクセスが必要な各ノードに対して定義されている

e. すべてのユーザー・アカウントはクラスタ・アクセス対応デバイス上のデフォルト・ディレクトリにログインする

2. アクセスが必要であると判断したユーザー・ノードからアクセス可能なディスクを決定します。次の場合に、最適なパスとデバイスを選択する必要があります。
   a. 高可用性が最優先である
   b. 高パフォーマンスが最優先である
3. 通常使用しているノードで障害が発生した場合、ユーザーが他のノードにログインできるかどうかを決定します。ノードで障害が発生した場合、次のいずれかの条件が真であることを確認する必要があります。
   a. ユーザーは他のノードまたは他の複数のノードにアカウントを持っている。
   b. すべてのノードは共通 SYSUAF.DAT ファイルを共有している。Alpha プロセッサと VAX プロセッサに異なる SYSUAF.DAT ファイルを使用して、異なるメモリー管理作業セットを割り当てられるようにする（SYSUAF プロセスの制限と割当ては Alpha プロセッサのほうが高いため）。
4. .ruj ファイルを配置する場所を次から選択して決めます。
   a. システム論理を定義して 1 つのディレクトリを参照
   b. クラスタ・アクセス対応ディスク上にデータベース・ユーザーのデフォルト・ディレクトリを作成
   c. クラスタ・アクセス対応ディスク上のユーザー・ディレクトリを参照するようにプロセス論理を定義
5. データベース・ファイルの設置場所を決定。データベース・ファイルはデータベースを使用するすべてのノードからアクセスできる必要があります。
6. SQL の CREATE DATABASE 文と ALTER DATABASE 文を発行して指定したデバイス上にデータベースを作成し、.rdb ファイルが存在するデバイス以外のデバイス上に .aij ファイルを作成します。◆

6.4 シングルノード・データベースを VMScluster データベースに変換

OpenVMS VAX OpenVMS Alpha

この項では、シングルノード・データベースを VMScluster 環境で効率よく動作するデータベースに変換する方法を説明します。6.3 項（6-20）でリストしたステップは、マルチファイル・データベースを作成する方法を示しており、この場合にも最後のステップを除いて当てはまります。次に示す方法に従うと、現行のシングルノード・データベースを複数のユーザーが利用できるようになります。

- RMU Backup コマンドと RMU Restore コマンドを使用します。
  この手順における RMU Backup と RMU Restore の使用方法の説明は、『Oracle Rdb7 Guide to Database Maintenance』を参照してください。迅速なため、この方法が好まれます。RMU Restore コマンドで Directory、Root、File および Snapshots 修飾子を指定する必要があります。『Oracle RMU Reference Manual』では、このコマンドに対する構文と修飾子について説明しています。
- あるいは、SQL の EXPORT 文や IMPORT 文を使用します。
  例 6-11（6-25）では、SQL の EXPORT 文と IMPORT 文を使用してシングルノード・データベースを VMScluster でアクセスできるデータベースに変換する方法を示します。この方法は、通常 RMU Backup や RMU Restore よりも実行時間がかかります。したがって、この特別な作業を実行するための、より優れた方法ではありません。SQL の EXPORT 文や IMPORT 文を使用するアプリケーションの詳細は、『Oracle Rdb7 Guide to Database Maintenance』を参照してください。この 2 つの文の構文と引数の詳細は、『Oracle Rdb7 SQL Reference Manual』を参照してください。

3.2 項（3-8）と 8.1.1 項（8-2）では、複数の記憶デバイスに渡って I/O 操作を分散することでデータベースのパフォーマンスを改善するための説明を記載しました。6.2.6 項（6-16）では、Oralce rdb ファイルの配置について説明します。

SQL の EXPORT 文と IMPORT 文を使用すると、次の処理を行うことができます。

- SQL の EXPORT 文を発行して、データベースの交換ファイルを作成します。
- SQL の IMPORT 文を発行して、共有ディスク・デバイスにデータベースとジャーナル・ファイルを分散させます。

データベースをシングルノードから VMScluster 環境へ変換する作業は、困難ではありません。しかし、この作業を始める前に代替手段が利用できるかどうかを十分に評価しておく必要があります。

例 6-11（6-25）の DB_DISK、DISK3、DISK4、AIJ_DISK1 および AIJ_DISK2 は、例 6-6（6-21）で定義されています。これらの論理名を使用して、mf_personnel データベース・ファイルを適切なディスク・デバイス上に配置します。例 6-11（6-25）は、シングルノード・データベースから VMScluster データベースへの変換に SQL の EXPORT 文および IMPORT 文を使用する方法を示しています。

例 6-11 SQL の EXPORT と IMPORT 文を使用してシングルノード・データベースを VMScluster データベースに変換

```
$ SQL
SQL> EXPORT DATABASE FILENAME DB_DISK:[DBS.ROOT]mf_personnel
  1> INTO DISK11:[DBS.EXP]mf_personnel.rbr WITH EXTENSIONS;
SQL>
SQL> IMPORT DATABASE FROM DISK11:[DBS.EXP]mf_personnel.rbr
  1> FILENAME DB_DISK:[DBS.ROOT]mf_personnel.rdb
```

```
  2> PATHNAME "SYS$COMMON:[HEADQUARTERS]mf_personnel"
  3> NUMBER OF USERS IS        50
  4> NUMBER OF BUFFERS IS      20
  5> NUMBER OF CLUSTER NODES IS 16
  6> NUMBER OF RECOVERY BUFFERS IS 20
  7> BUFFER SIZE IS             6 BLOCKS
  8> SNAPSHOT IS ENABLED
  9> DICTIONARY IS REQUIRED
 10>
 11> CREATE STORAGE AREA DEPARTMENTS
 12>   FILENAME DISK3:[PERS.STOR]departments.rda
 13> ALLOCATION IS 25 PAGES
 14> PAGE FORMAT IS MIXED
 15> SNAPSHOT FILENAME DISK4:[PERS.STOR]departments.snp
 16> SNAPSHOT ALLOCATION IS 10 PAGES
 17> END IMPORT;
SQL> FINISH;
SQL>
SQL> ALTER DATABASE 'DB_DISK:[DBS.ROOT]mf_personnel'
  1> ADD JOURNAL AIJ_ONE
  2> FILENAME 'AIJ_DISK1:[PERS.AIJ]aij1.aij'
  3> ADD JOURNAL AIJ_TWO
  4> FILENAME 'AIJ_DISK2:[PERS.AIJ]aij2.aij';
SQL>
```

次の要約では、既存のシングルノード・データベースをクラスタアクセス対応データベースに変換するために必要なステップを説明します。

1. SQL の EXPORT 文を発行して、データベースの交換ファイルを作成します。
2. データベース・ファイルの設置場所を決定します。
   rdb、.rda、.snp、.ruj および .aij ファイルを含むすべての Oracle Rdb ファイルが、データベースへのアクセスが必要なすべてのノードからアクセス可能なデバイス上にあることを確認します。
3. SQL の IMPORT 文を発行して、クラスタアクセス対応ディスク・デバイスにデータベースをリストアし、データベースとジャーナル・ファイルを分散します。SQL の IMPORT 文句を使用して、データベース・パラメータを再度定義します。
4. 新しいデータベースをテストして、データベース定義がデータベースそれ自身とリポジトリの両方から利用でき、データベースとそのデータが各ノードからアクセスできることを確認します。
5. 古いデータベースを削除します。◆

6.5 自動リカバリ手順

OpenVMS VAX OpenVMS Alpha

VMScluster 構成内のノードで障害が発生した場合、Oracle Rdb はデータベースの自動リカバリを実行して、そのデータベースを矛盾のない状態にします。

VMScluster 環境における Oracle Rdb データベースの通常の操作状態には、データベースにアクセスできる各ノード上で動作する Oracle Rdb モニター・プロセスが含まれています。ノードで障害が発生した場合、そのノード上のすべてのプロセスは終了します。終了したプロセスがデータベースにアクセスしていた場合、OpenVMS ロック・マネージャは、1 人以上のユーザーがデータベースにアクセスしている生存ノード上のモニターに対して（デッドマンと呼ばれる）特別なロックを与えます。生存ノード上のモニターが、障害が発生したノードからアクセスされたデータベースに対するデッドマンロックを与えられたとき、その生存ノードは VMScluster 内のすべてのノードに対するデータベース内のすべてのアクティビティを一時停止します。デッドマンロックを与えられた生存ノードは、データベース・リカバリ処理（DBR プロセス）を開始します。

DBR プロセスは、データベースを矛盾のない状態に戻します。データベース・リカバリ手順は、高速コミット処理がデータベースに対して有効になっているかどうかによって異なります。ノードで障害が発生し、障害が発生したノード上でアクセスされていたデータベースは高速コミット処理が有効な場合、DBR プロセスは、まず .aij ファイルを使用して、各ユーザーの最後のチェックポイント以降にデータベース内でコミットされたすべてのトランザクションをリカバリします。このコミットされたトランザクションのリカバリは、再実行操作と呼ばれています。そして DBR プロセスは、.ruj ファイルに記録されているコミットされていないトランザクションをロールバックします。このコミットされていないトランザクションのロールバックは、取消し操作と呼ばれています。

ノードで障害が発生して、障害が発生したノードからアクセスされていたデータベースの高速コミット処理が有効でない場合、DBR プロセスは、.ruj ファイルに記録されているコミットされていないトランザクションをロールバックします。しかし、コミットされた更新はすでにデータベース領域ファイルに書き込まれているので、再実行操作は必要ありません。

生存ノード上のモニター・プロセスがデータベースで必要なすべての .ruj ファイルにアクセスできなければ、このデータベースはシャットダウンされます。リカバリを完了するには、データベースは、.ruj ファイルにアクセスできるノードからリカバリする必要があります。

データベースがリカバリされた後、障害が発生したノードのユーザーは、VMScluster 構成内の生存ノードにログインでき、データベースの使用を再開できるようになります（ユーザーが生存ノードにアカウントを持っている場合）。

このシナリオは、共有可能なディスク・デバイス上でのデータベースおよびジャーナル・ファイルの作成と維持の重要性を説明しています。

VMScluster 上に Oracle Rdb の 2 つ以上のバージョンがインストールされている場合、各ノード上にインストールされている Oracle Rdb の各バージョンに対して、バージョン固有のモニター・プロセスが存在することに注意してください。Oracle Rdb 複数バージョン環境では、各ノードには 2 つ以上のモニター・プロセスがあります。複数バージョン環境内で

ノードに障害が発生すると、インストールされている Oracle Rdb の各バージョンに対してリカバリ手順が発生します。

障害が発生したノードだけがデータベースへアクセスしていた場合、リカバリ手順はシングルノード・データベースの場合と同じになります。リカバリ処理は、次回プロセスがデータベースに接続されるときに起動します。Oracle Rdb は、HSC サブシステムの障害をプロセッサの障害とは異なる方法で取り扱います。ユーザー・プロセスが異常終了していないので、リカバリ手順は起動されません。データベースへの影響は、VMScluster 環境の構成によって異なります。

- HSC ディスクがデュアルパスであれば、OpenVMS オペレーティング・システムは自動的にデータベース・アクセス・パスを代替 HSC サブシステムに切り替えます。データベースへのアクセスは、中断されません。
- HSC ディスクがシングルパスであって、データベース・ファイルが HSC ディスク上にあれば、データベースへの接続は、HSC サブシステムが操作可能になるまで中断されています。次にプロセスがデータベースに接続したとき、リカバリ手順が開始されます。

6.5.1 項（6-28）と 6.5.2 項（6-29）では、Performance Monitor のデータベース・リカバリ画面について説明しています。◆

6.5.1 Performance Monitor の「Recovery Statistics」画面

OpenVMS VAX OpenVMS Alpha

Performance Monitor は、種々のリカバリ・フェーズを識別して、各フェースの完了までにかかった時間の情報を表示する「Recovery Statistics」画面を提供します。

「Recovery Statistics」画面は、異常なプロセス障害の数が多すぎないか識別するのに便利です。さらに、この画面は、実行時のパフォーマンスを最大にして、リカバリ停止時間を最小に設定する正しいデータベース属性とパラメータ設定を決定する場合に便利です。

「Recovery Statistics」画面には、個々のプロセスのリカバリだけでなく、障害が発生したすべてのプロセスのリカバリに対するグローバルな情報が表示されることに注意してください。

「Recovery Statistics」画面は、「Journaling Information」サブメニューから表示します。次の例に、「Recovery Statistics」画面を示します。

```
Node: TRIXIE         Oracle Rdb V7.0-00 Performance Monitor  3-FEB-1996 08:04:27
Rate: 1.00 Second             Recovery Statistics           Elapsed: 00:26:45.95
Page: 1 of 1        RDBVMS_USER1:[LOGAN.V70]PERSONNEL.RDB;1         Mode: Online
-------------------------------------------------------------------------------
statistic.........      rate.per.second.............. total....... average......
name..............      max..... cur..... avg....... count....... per.trans....
process attaches               0        0        0.0           4           1.0
process failures               0        0        0.0           1           0.2
DB freeze len x100             0        0        0.4         684         171.0
Tx REDO count                  0        0        0.0           1           0.2
  redo time x100               0        0        0.2         389          97.2
Tx UNDO count                  0        0        0.0           1           0.2
  undo time x100               0        0        0.0           9           2.2
No UNDO needed                 0        0        0.0           0           0.0
Tx committed                   0        0        0.0           0           0.0
Tx rolled back                 0        0        0.0           0           0.0
No resolve needed              0        0        0.0           1           0.2
AIJ recover x100               0        0        0.0           6           1.5
GB recover x100                0        0        0.0           0           0.0
Cache recover x100             0        0        0.0           0           0.0
RUJ file reads                 0        0        0.0           1           0.2
AIJ file reads                 0        0        0.0           9           2.2
-------------------------------------------------------------------------------
Exit Graph Help Menu Options Pause Reset Set_rate Time_plot Unreset Write X_plot
```

この画面の説明は、Performance Monitor のヘルプを参照してください。◆

6.5.2 Performance Monitor の「DBR Activity」画面

OpenVMS VAX OpenVMS Alpha

Performance Monitor には、現行ノード上のアクティブ DBR プロセスの情報を取得できる「DBR Activity」画面があります。「DBR Activity」画面は、Performance Monitor が実行されるノード上でアクティブな各 DBR プロセスに対して 1 行の情報を表示します。ノード上にアクティブな DBR プロセスがなければ、画面は空になることに注意してください。

「DBR Activity」画面は、「Process Information」サブメニューから表示します。

この画面を使用しない場合には、DBR プロセスが実行中であるかどうかを判断するためにユーザーが利用できる唯一の方法は、RMU Show Users ユーティリティを使用することです。しかし、RMU Show Users ユーティリティは、DBR プロセスが実行中であることのみを示し、リカバリ操作で DBR が行っている内容が何であるかは示しません。次の例は、「DBR Activity」画面を示しています。

```
Node: TRIXIE         Oracle Rdb V7.0-00 Performance Monitor 28-MAY-1996 16:01:59
Rate: 3.00 Seconds                DBR Activity              Elapsed: 05:41:05.72
Page: 1 of 1         SQL_DISK1:[RICK.V70]MF_PERSONNEL.RDB;1         Mode: Online
-------------------------------------------------------------------------------
Process.ID Activity... VBN.... Operation.......................... Lock.ID.
12345678:1 TX redo             Reading page 1:2
-------------------------------------------------------------------------------
Exit Help LockID Menu >next_page <prev_page Set_rate Write Zoom !
```

この画面の説明は、Performance Monitor のヘルプを参照してください。◆

6.6 データベースのメンテナンスと監視

OpenVMS VAX OpenVMS Alpha

この項では、VMScluster 環境における Oracle Rdb データベースの通常の操作とメンテナンスについて説明します。

VMScluster 環境の Oracle Rdb データベースで行われるほとんどの操作は、シングルノード環境の Oracle Rdb データベースで行われる操作と同じです。◆

6.6.1 ローカル・エリア VMScluster 構成についての検討事項

OpenVMS VAX OpenVMS Alpha

一般的なローカル・エリア VMScluster 構成は、大きなプロセッサ 1 つといくつかの小さなプロセッサから構成されています。構成内の小さいほうの各プロセッサは、アプリケーションを実行できるだけの十分なメモリーを持っていることを確認する必要があります。データベースの作成には、他のアプリケーションより多くのメモリーが必要なので、データベース作成のタスク用に 1 つのプロセッサを決めることになると考えられます。ページングやスワッピング・ファイルには、ローカル・エリア VMScluster システム内のローカル・プロセッサを使用することもあります。

次に示す AUTOGEN パラメータを注意して監視してください。

- PAGEDYN
- NPAGEDYN
- GBLSECTIONS
- GBLPAGES

これらの各パラメータの値が、アプリケーションに対する対話型ユーザーの数や、予定されているバッチ・ジョブの負荷を収納できる十分な大きさであることを確認してください。

ローカル・エリア VMScluster 環境内の単一または複数のブート・ノードは、サテライト・ノードに対するディスク・サーバーとして動作するため、パフォーマンスのペナルティを負います。**サテライト**・ノードは、ブート・ノードのシステム・ディスクからリモートで起動する任意のプロセッサです。一般的に、これらのノードはクラスタのリソースを消費しますが、ディスクやバッチ処理を行うリソースを提供するようにも設定できます。

ローカル・エリア VMScluster 内のブート・ノードで障害が発生すると、クラスタ操作はこのノードがクラスタに再結合するまで一時停止します。この条件は通常であり、共有クラスタ・リソースの整合性を保証します。◆

6.6.2 データベースの監視

OpenVMS VAX OpenVMS Alpha

クラスタ内の任意のノードからデータベースについての情報を表示できます。RMU Show Users コマンドと RMU Show System コマンドは、コマンドが発行されたノード上のユーザーについての情報を表示します。

データベースについての情報を表示するには、次のコマンドを使用します。

- RMU Analyze Areas
- RMU Analyze Lareas
- RMU Analyze Indexes
- RMU Analyze Placement

ノード上のユーザーについての情報を表示するには、次のコマンドを使用します。

- RMU Show Users
- RMU Show System

これらのコマンドは、一度に 1 つのノードでしか実行できません。この制限は、VMScluster 環境内に Oracle Rdb を実装するために必要な分散ルート・ファイル・アクセスと複数のモニター・プロセスによるものです。そのデータベースに対するアクティビティの完全な状況を得るには、データベース・ユーザーのプロセスが存在する VMScluster システム内の各ノード上で RMU Show Users および RMU Show System コマンドを実行する必要があります。これは、これらのコマンドをコマンド・プロシージャに投入して実行します。これらのプロシージャは、ユーザーがデータベースに対するアクセスを持つ各ノードのバッチ・キューで同時にサブミットできます。SQL の SET OUTPUT full-file-spec 文を使用して、シングル・ディレクトリ内のディスク・ファイルに直接出力します。それから、ファイルの内容を表示または印刷します。

RMU Dump Users コマンドを使用すると、クラスタに関するすべてのデータベース・ユーザー・プロセスについての情報を表示できます。

```
$ RMU/DUMP/USERS mf_personnel
```

Performance Monitor を使用して、データベース・アクティビティを監視することもできます。データベースのモニタリングの詳細は、2.2 項（2-15）を参照してください。◆

7

チューニングの概念と方法論

システムのチューニングは、タイムシェアリング環境では新しい概念ではありません。十分に調整されたコンピュータ・システムでは、リソースは、マシンが本来意図した作業負荷に集中的に配分されています。この章では、システムのチューニングがデータベースのパフォーマンスに与える影響を調査します。この章では、モデルとして製造業環境を使用しますが、開発されたデータベース・チューニング概念は、すべてのデータベース環境に適用されます。

Computer Integrated Manufacturing（CIM）環境では、情報を迅速に保存、アクセスする能力は、現場のプロセス要件にとって非常に重要です。さらに、製造プラントでは、常により強力なハードウェアを整備して応答時間を改善することはできません。この状況では、データベースのパフォーマンスを最適化するためにシステムをチューニングすることで、全体の運用に効果を及ぼし、問題に対する低コストのソリューションを提供できます。

7.1 チューニングとは何か

チューニングとは、システムまたはデータベース・パラメータを調整して中央処理装置（CPU）、メモリーまたは入出力（I/O）の容量を追加せずに、対象の作業負荷に対してリソース全体の使用を最適な状態にすることです。環境にメモリーや他のディスク・ドライブを追加するのは、チューニングではないことに注意してください。チューニングとは、利用可能なリソースで対処可能にすることです。その核心は、どの作業負荷がこのリソースを使用しているか、またこれらの作業負荷のうちどれが最も重要かを理解することです。次に、最も重要な作業負荷に対して、必要なときに常にリソースを使用できるようにシステムをセットアップします。

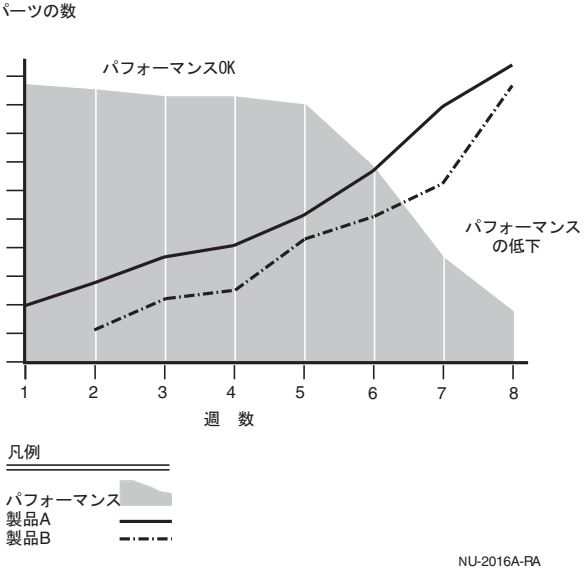
7.2 チューニングのタイミングの決定

チューニングは、環境内で継続的に行う通常の処理ではありません。そのかわり、システムのモニタリングは定期的に行う必要があります。チューニングは、通常、応答時間が低下した場合やシステムのモニタリングがリソース不足を示したときに行います。

チューニングが必要なタイミングを決めるには、作業負荷とリソースの利用状況を理解する必要があります。アプリケーションの応答が予測よりも悪い場合、重要なリソースへの要求によって、問題が発生しています。さらに調査する必要があります。

時間の経過に従って作業負荷が変化してきたら、再度チューニングを考える必要があります。この状況は、生産量の変動や製品の増加によって作業負荷が徐々に変動する製造環境ではよく発生します。10% の製品量では適切に動作するシステムとデータベース・パラメータが、より高い割合では同じように動作しないことがあります。パフォーマンスの低下が顕著になったら、調査してください。図 7-1（7-3）では、この状況を強調しています。

図 7-1 作業負荷の変動によるチューニング



ハードウェア容量の増加は、環境の調整を行う最も一般的な理由です。たとえば、マシンのメモリー容量を 2 倍にすると、古いハードウェア構成に対する作業セットは最適ではなくなります。追加メモリーを利用するために作業セットを調整すると、多くのメモリー容量が必要なアプリケーションのパフォーマンスは劇的に向上します。同様に、この構成にディスクを追加し、データベース・スキーマを変更して追加デバイスにデータを分散させると、I/O や競合問題を防ぐことができます。チューニングを行うと、新しい容量の利点を最大限に生かして利用できます。

7.3 リソースのタイプ

Oracle Rdb は、排他と共有の 2 つのタイプのリソースを使用します。

幸いなことに、Oracle Rdb では、作業負荷の要求に従って排他的に使用しなければならないリソースはごく少数です。排他的リソースを使用しなければならない場合、重要性が低い作業負荷によって重要な作業負荷がリソースへのアクセスを阻止され、重要な作業が遅延することがあります。これによって、パフォーマンスが低下することがあります。たとえば、テーブルに対してインデックスを定義すると、このテーブル（リソース）は定義者が排他的に使用しなければなりません。これによって、テーブルへのアクセスが必要な作業負荷の競合が増加します。

もう 1 つのタイプのリソースは共有リソースです。これは、データベース環境ではより一般的に見られるリソースです。このリソース・タイプに関連する通常の問題は、リソースが共

有過多になることです。この問題を説明するために、りんごを使って環境内のリソースを表現してみましょう。りんごをいくつかにスライスして、これらの小片を何人か（作業負荷を表しています）に分けた場合、小片では空腹は満たされないと考えられます。彼らは、他の小片が得られるのを待つことになります。いくつかの作業負荷が他よりも重要性が高ければ、コンピュータ・システムを調整して、これらのより重要な作業負荷に対して必要な割合のリソースをすぐに与えられるように保証したいと考えるでしょう。

チューニングは、提示された環境変化の結果に対して影響をおよぼす多くの要因を持つ複雑な処理です。7.5 項（7-5）では、この複雑さを処理するデータベースのチューニング方法論を説明します。

7.4 データベース・アプリケーションのサンプル

この章では、例を使用してチューニングの概念を説明します。これらの例の多くは、製造リソース計画（MRP II）データベース・アプリケーションである PRODUCT_DB に基づいています。例を読みながら、PRODUCT_DB アプリケーションの設計をイメージしてください。

MRP II アプリケーションは、製造計画および材料や容量資源の制御に適用される種々のビジネス機能を管理、実行するために設計されたシステムです。MRP II は、トランザクション駆動システムであり、レコードの正確さに強く依存しています。このシステムは、システムの応答と柔軟性に重い要求を行うユーザーに厳格な規則を要求します。

PRODUCT_DB などの一般的な MRP II アプリケーションには、製造プラントの情報が含まれています。製品の品質を保証するため、テスト中に生成された異なるパラメータ・データとともに組み立てられたユニットの異なるサブコンポーネントの追跡に必要な情報（材料の目録）が含まれています。このデータベース上で動作するアプリケーションは、上級開発技術者が製造プロセスを洗練するために使用し、品質技術者が優れた製品を作成するために使用し、また製造者が仕事場での情報を追跡するために使用します。

表 7-1（7-4）は、PRODUCT_DB データベース内の各テーブルを簡単に説明しています。

表 7-1 PRODUCT_DB データベース・テーブル

テーブル名	説明
LOT_INFORMATION	製造現場で材料を追跡。
SHIPPING_INFORMATION	完成した製品の出荷日時と出荷先を追跡。
PARTS	パーツとロット名をリンク。
PART_DEFECT	部品の欠陥の理由を識別。
DEFECT_DESCRIPTIONS	欠陥コードを説明に変換。ベンダーにこの説明を送り返すことで、パーツが拒否された理由を識別。
LOT_DATA	ロットに関連するパラメータ・テスト・データを保存。

表 7-1 PRODUCT_DB データベース・テーブル（続き）

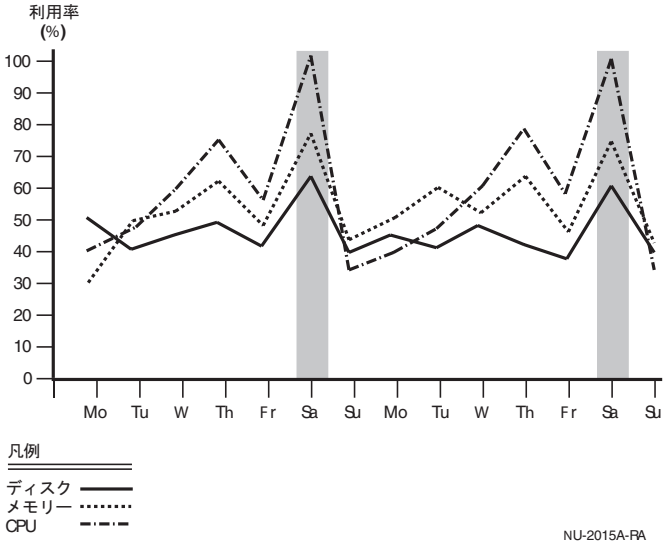
テーブル名	説明
PROCESS_SPECIFICATIONS	指定されたプロセス・ステップで製品に対するテスト制限を制御。

7.5 チューニングの方法論

この項では、データベースのチューニングの方法論について説明します。ここでは、データベースのチューニング問題に対処するための 6 つのステップを示します。潜在的な問題が認識された段階では、多くの場合、問題の原因は不明であり、問題に関連する情報は不明瞭です。たとえば、次のシナリオを見てください。

MRP II アプリケーションでデータ入力を行っているユーザーは、数か月の周期でパフォーマンスが低下していることを認識しています。ユーザーは、データベースの専門家であるシステム・マネージャに不満を訴えます。どこから開始しますか。

図 7-2 リソースの使用状況を分析



1. この環境で実行されている作業負荷を理解します。
   最初のステップは、環境内の作業負荷を理解することです。およそ 2 秒で応答していたアプリケーションが、急に 5 秒で応答するようになってしまったのか。最近環境が変更されたか。（たとえば、新しいバージョンのソフトウェアや階層化製品がインストールされたか。）この問題が発生するのは、図 7-2（7-5）で示すように 1 日の決まった時間

か、または 1 週間の決まった曜日か。
このプロセスのこのフェーズの学習では、パフォーマンス・レポートを利用できると便利です。この情報の収集に OpenVMS Monitor ユーティリティ（MONITOR）などのツールが使用できます。リソースの利用率がいつ最大または最小になるかを理解するには、システム・リソースのグループを毎日見ることから始めます。たとえば、製造環境では、リソースの利用率は週日の間少しずつ上昇し、出荷分担の日にピークをむかえます。これを理解すると、パフォーマンス問題の診断を利用率が最小である月曜日に行おうとはしないでしょう。そのかわり、チューニングの分析をリソースがより緊迫していて、分析がより効果的に行えそうな土曜日に行うように計画します。（制限されたリソースはピーク期間中にさらにはっきりします。）四半期の終わりの出荷期限と年末の出荷期限時に利用率が上がることに気づくでしょう。
トランザクションと量分析情報（トランザクション量、トランザクションのタイプ、トランザクションの長さ）が利用できれば、便利です。環境で動作するアプリケーションを理解するにつれて、問題の原因に対する洞察力が高まります。

2. ハードウェアの問題をチェックします。
この問題を経験しているユーザーは、自分たちが作業しているコンピューティング環境を認識していないか、理解していない可能性があります。チューニングを始める前に、システム上に主要なハードウェア問題がないか確認してください。クラスタ環境で実行している場合、通常の数のマシンが動作しているかを確認してください。（マシンがオフラインであれば、応答が遅いというユーザーの感覚に原因がある可能性があります）

OpenVMS VAX OpenVMS Alpha

DIGITAL Command Language（DCL）の SHOW ERROR コマンドを使用して、環境や作業負荷に影響を与えている問題がないかを確認します。たとえば、データベース・ディスク上に多く他のエラーがありませんか。
しかし、SHOW ERROR コマンドはすべてのハードウェア問題を示すわけではありません。たとえば、多くの数のデータベース・ディスクが複数の HSC サブシステムに渡って分散されていて、1 つの HSC で障害が発生したら、すべてのデータベース・ディスクが同じ HSC に切り替わります。このように 1 つの HSC サブシステムに切り替わると、動作が遅くなります。この問題を診断する 1 つの方法は、DCL SHOW DEVICES コマンドを使用してどの HSC サブシステム上でディスクがオンになっているかを判断することです。（ディスクが 1 つの HSC 上にあれば、HSC 名はデバイス名の後のカッコ内に表示されます。）◆
システム・マネージャを使用してハードウェア構成が最近変更されていないことを確認することもよい方法です。代用品が見つかるまで、欠陥があるハードウェアをシステムから切り離すことは可能です。オリジナルの構成に容量を追加することに慣れているユーザーは、応答時間の問題による変化に気づいているでしょう。

3. 問題のある領域に焦点を当てます。
問題がある領域に焦点を当てるため、I/O、メモリー、CPU およびロックの利用率の傾向を注意して検査します。この環境における作業負荷の通常の操作を反映していないデータから結論を出さないようにしてください。数週間分の情報を見るようにしてください。数週間分のデータが入手できなければ、持っているデータを使用します。少ないサンプルを使用するリスクは、結果を導き出すベースにパフォーマンスの低下の原因が見られなかったオフピーク期間が含まれる可能性があるためです。65% 以上がビジーで

あるリソースではパフォーマンスの低下が始まっていることに注意してください。目的は、容量があり、他よりも利用率の低いリソースに作業を広げることです。

4. 問題の原因を判断します。
問題を 4 つの基本領域（I/O、メモリー、CPU またはロック）の 1 つに分類したら、ボトルネックを取り除くことが可能なオプションを判断するため、システム・レベル・パラメータ、データベース・パラメータまたはアプリケーションそれ自体を深く観察します。

5. 実行可能なソリューションを選択して、変更を行います。
このステップには、利用できる最高のソリューションの選択と環境に適用する変更が含まれています。ソリューションを選択するときに考慮する必要がある重要な点には、コスト、実装時間、リスクおよび改良の可能性が含まれます。

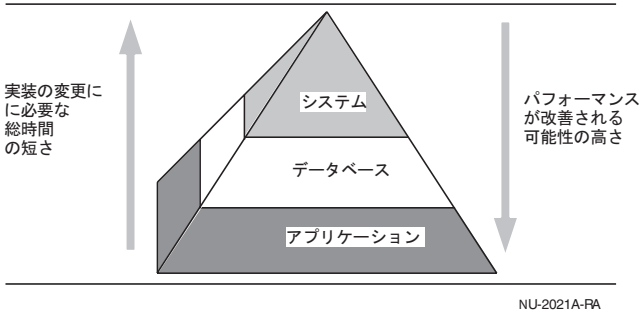
6. 結果の監視。
変更が行われた後、状況の改良を確認するため、この環境を監視する必要があります。

7.6 チューニング対象の決定

チューニング方法論を詳細に調査する前に、チューニングする対象を決める必要があります。7.5 項（7-5）では、I/O、メモリー、CPU、ロックなどのリソースをチューニングする場合、特に 4 つの基本的な領域に焦点を当てる必要があることを強調しています。これらのリソースのうちの 1 つの利用率が 65% 以上であれば、常にパフォーマンスが低下する可能性が存在します。

焦点を当てる領域を選択したら、ソリューションのチューニングを決定する前にいくつかの要因を考慮する必要があります。図 7-3（7-7）は、この概念を説明しています。

図 7-3 可能性を改善する領域



7.6.1 システムのチューニング

図 7-3（7-7）は、環境の構成要素による分類を示しています。パフォーマンスの改善は、**システム、データベース**および**アプリケーション**の 3 つのコンポーネントに集約できます。この分類によって、リスクやコストなどの問題が処理できます。

OpenVMS VAX OpenVMS Alpha システム・チューニングの構成要素は、OpenVMS System Generator ユーティリティ（SYSGEN）、OpenVMS Authorize ユーティリティ（AUTHORIZE）、および RDM$BIND_BUFFERS などのその他の動的パラメータで構成されています。◆

図 7-3（7-7）のピラミッドを見ると、このシステムは最も小さい領域であることがわかります。つまり、システムのチューニングによって得られるパフォーマンスに対する可能性は最小です。

システム・レベルのチューニングによってパフォーマンスが改善される可能性は他の領域に比べて小さいものですが、このレベルのチューニング・セッションを開始することでなんらかの利点が得られます。まず最初に、変更を実装するために必要な時間がわずかです。いくつかの影響を与えるパラメータは動的であり、すぐに変更できます。その他のパラメータは、次のログインまたは次にシステムをリブートすると、有効になります。その他の利点は、リスクがより小さいことです。変更はほとんどすぐに取り消すことができます。システムの停止時間が製品の量に直接影響を与える製造環境では、実装フェーズの間に、製造に影響を与えることなくパフォーマンスを改善するリスクの低いチューニングの変更は価値のある拡張です。変更を加えるためにプラントをシャットダウンする必要があれば、好ましいとはいえません。

システム・レベルのチューニングのもうひとつの利点は、コストです。高価なプログラマやデータベース管理者リソースは、このレベルでの変更には通常必要ありません。ここで遭遇する問題は、一般的に 1 つのアプリケーションよりもさらにグローバルなので、内部のアプリケーション・コードを理解する必要はありません。これは、決定的なポイントになります。システム・レベルのチューニングは、アプリケーションに特有の情報があまり必要ではないので、それほど複雑ではありません。

7.6.2 データベースのチューニング

データベース・チューニングの構成要素には、ファイルの位置の指定、データベース・バッファの数と大きさおよび初期ファイル割当てなどの機能を実行するデータベース定義言語があります。この構成要素には、アプリケーション・コードを変更することなく、問題の解決に使用できるすべての非システム・パラメータが含まれます。図 7-3（7-7）のデータベース領域は、システムとアプリケーション領域の間に入っています。これは、データベース・レベルでのチューニングは、システムでのチューニングよりもさらに大きな効果をもたらす可能性がありますが、アプリケーションの変更よりも効果は小さいことを示しています。

データベース・レベルでのパラメータへの変更は、システム・レベルでの変更に比べると一般的に複雑です。データとそのデータがどのように使用されるかに対して、より明白な知識が必要です。通常、データベースのチューニングは、データベース管理者を通して調整、実装する必要があります。これは、システム・レベルでの変更よりも遅延が長くなることを示しています。

ほとんどのデータベース・パラメータは動的に変更できません。変更の多くは、データベースに対する排他的アクセスが必要です。その他は、変更から最大の効果が得られるように、データベースをエクスポートして、その後でインポートする必要があります。この欠点は、リスクとコストが高いことです。データベースの動作する環境が年中無休の稼働時間を要求

している場合、データベース・レベルでのチューニングには避けられない影響があります。プラント休止の費用が要因に加えられるので、コストはさらに高くなります。データベースをオンラインに戻すときに予期せぬ遅れが発生する可能性があるので、リスクはさらに高くなります。オンラインで行う変更のリスクを減らすため、しばしば開発環境が設定されます。提案されたすべてのデータベースへの変更は、本番環境に実装される前に開発データベース内でテストされます。開発用のデータベースのほとんどは本番データベースと同じ大きさではないので、リスクが完全に取り除かれるわけではありません。開発データベースを使用するかどうかを決定する場合、開発データベースを設定するコスト、必要なハードウェアのコストおよびすべての変更を 2 度行うコストと予期せぬ問題が発生して長期間データベースがオフラインになった場合のコストを比較検討します。

7.6.3 アプリケーションのチューニング

アプリケーション・チューニングの構成要素には、アプリケーション・コード、トランザクション情報およびユーザーとの対話が含まれています。アプリケーション領域は、図 7-3（7-7）に示した 3 つの領域で最も大きなものであり、ピラミッドの基盤になっています。これは、アプリケーション・レベルのチューニングはパフォーマンス全体に最大の効果をもたらす可能性があることを示しています。

アプリケーション・レベルでの変更は、アプリケーション・コード、トランザクションのタイプとサイズおよびアプリケーションが動作している環境などに対する詳細な知識が必要であり、最も複雑です。大規模な環境では、変更に対して複数のアプリケーションとプログラマが関連してくる可能性があります。アプリケーション・レベルの変更は、他のレベルの実装に比べてさらに時間がかかります。

アプリケーション・レベルの一般的な変更では、アプリケーション・コードの変更と再構築が必要です。アプリケーションのコードを変更すると、新しいバグが入り込む可能性があります。これは、リスクとコストに対してあらゆる影響を与えます。テスト環境に対して 7.6.2 項（7-8）で説明した議論と同じ議論がここで適用できます。次の段落では、このレベルで解決できる問題の例を示します。

ユーザーの応答時間の問題が、1 日の特定の時間内でのみ発生し、それが PARTS テーブルでの過剰ロックによって発生することがわかったと仮定します。Performance Monitor を使用して、多くのユーザーがデータベース・ページが解放されるのを待っていると「Stall Messages」画面から判断します。問題の根本である原因を判断するため、実行中のアプリケーションのプロセス ID を追跡すると、実行中のアプリケーションの 1 つでオペレータが「Modify Part」画面で中断していることを見つけます。アプリケーション・コードを眺めてみると、書込みトランザクションの途中でユーザーの入力が必要になるようにアプリケーションが作成されています。ロック問題を防ぐ方法の 1 つは、トランザクションが開始される前に必要な情報をユーザーから得るように、アプリケーションのコードを書き変えることです。これは、実際の書込みトランザクションをできるだけ短くし、競合を減らします。

8

データベース・リソースのボトルネックの診断

データベース・アプリケーションは、一般にI/O集中型なので、この章では次の順番でリソースを分析するようにお薦めします。

- I/O
- メモリー
- CPU
- ロッキング

この章の各デシジョン・ツリーは、特定のパフォーマンス問題を分離、識別、分析および解決するために使用する計画されたアプローチを提供します。たとえば、図8-1（8-2）は、リソースのボトルネックを分析する1つのアプローチを示しています。このマニュアルのすべてのデシジョン・ツリーでは、ソリューションは太い境界線のボックスで表示されます。「へ進む」があるボックスは、そのラベルのついたデシジョン・ツリーを記述した図に進むことを意味します。残りのボックスは、移行過程のもので、ソリューションを導き出すために特定の質問を行います。他のデシジョン・ツリーも、これらの同じ規則に従います。それぞれの図は、低いリスクで容易に実装できるチューニング・ソリューションを推奨することを目的にしています。デシジョン・ツリーは、その他のより困難なオプションを考慮する前に、可能であればいつでもリスクの低いソリューションに到達するように設定されています。

注意：この章を読む際には、この章のRMU出力や他の場所ではリレーションという用語で表現している意味を、この章の本文ではSQL用語のテーブルを使用して表現していることに注意してください。

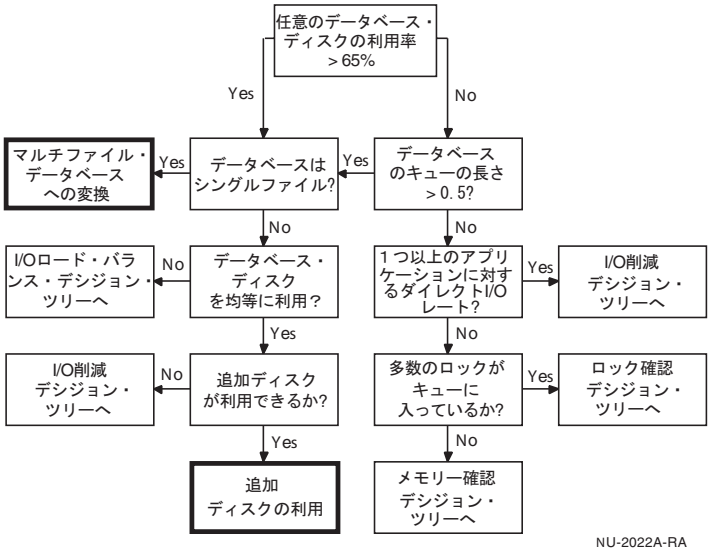
8.1 I/O リソースの分析

I/O リソースを精査する場合には、I/O リソースのボトルネックの検出、I/O ロードの均衡化および I/O 操作の削減に専念する必要があります。8.1.1 項（8-2）、8.1.2 項（8-4）および 8.1.3 項（8-15）には、これらの作業を実行する方法の詳細が説明されています。

8.1.1 I/O リソースのボトルネックを検出

図 8-1（8-2）に示すように、可能性がある I/O リソースのボトルネックに対してシステムの調査を開始する最初の場所は、ディスク・ドライブです。利用率が 65% を超えると、パフォーマンスはある程度の影響を受けます。

図 8-1 I/O リソースのボトルネックに関するチェックを行うデシジョン・ツリー



シングルファイル・データベースを持っていて、データベースの複雑さを低減するよりもパフォーマンスの方に関心がある場合、最初のステップはこのデータベースをマルチファイル・データベースに変更することでしょう。データベースをマルチファイル・データベースに変更する価値があるかどうかを見るために、シングルファイル・データベースのトランザクションに対するおよその最大スループットを決定します。シングルファイル・データベースで、指定されたトランザクションに対する最大スループットを決定する式を例 8-1（8-3）に示します。

例 8-1 シングルファイル・データベース内のトランザクションで可能なスループットを決定する式

$$\text{Transactions per Second} = \frac{\text{Disk I/Os Performed per Second}}{N + L}$$

シングルファイル・データベース内のトランザクションに対して可能な最大スループットを決定するには、最初にデータベース・ディスクが実行可能な 1 秒あたりのおよその I/O 回数を知っておく必要があります。例 8-2 (8-3) では、このディスクは 1 秒間に 30 回の I/O が実行できると仮定しています。2 番目にトランザクション（変数 N に対する例では値 7）が実行する必要がある I/O 回数と、トランザクション用に .rdb、.ruj および .aij ファイル（変数 L に対する例では値 3）に書き込まれる I/O 回数を知る必要があります。例では、トランザクションに対して可能な最大スループットは、例 8-2 (8-3) で示したように、1 秒あたり 3 トランザクション（tps）です。

例 8-2 シングルファイル・データベース内で与えられたトランザクションに対して可能なスループットを決定する

$$3\ \text{Transactions per Second} = \frac{30}{7 + 3}$$

トランザクションによって、7 回の I/O 操作が行われる場合、シングルファイル・データベース内で期待できる最高のスループットは、1 秒あたり 3 トランザクション（tps）です。スループットを改善したければ、データベースをマルチファイル・データベースに変更する必要があります。データベースをマルチファイルにしただけでは、よいパフォーマンスが得られるわけではないことに注意してください。むしろ、マルチファイル・オプションによって、チューニング・プロセスの間により多くのオプションを調査する可能性が得られます。マルチファイル・データベース内でボトルネックが発生している場合、ボトルネックがある場所を分析および理解して、ボトルネックを取り除くために適切な変更を行う必要があります。

データベースがマルチファイル・データベースであれば、チューニング・オプションの 1 つに、1 つのデバイス上の I/O 競合を低減するために複数のディスクに渡って分散させる方法があります。データベース・ディスクの 1 台の利用率がまだ高いことがわかれば、いくつかのオプションがあります。追加ディスクが利用できれば、これを使用してデータベースをさらに分散させます。これが選択できなければ、既存のディスク・リソースに渡る I/O ロードのバランスを改善する必要があります。データベースを分散または再調整する場合、データベースのルートと .aij ファイルが高トランザクション環境でデータベース・ホット・スポットになり得ることを思い出してください。ルート・ファイル・ホット・スポットは、4.1.5 項 (4-89) と 4.1.5.3 項 (4-98) で説明したように、高速コミット・トランザクション処理とジャーナル最適化を適切に使用して緩和できます。また、ルート・ファイルと .aij ファイルは、それ自身独自のディスク上に存在する必要があります。ルート・ファイルを独自のディスク上に配置できない場合には、利用率が高くないディスク上に置いてください。異なるタイプのディスク・デバイスを持つ環境で操作している場合、ルート・ファイルと .aij ファイルを最も高速のデバイス上に置く必要があります。

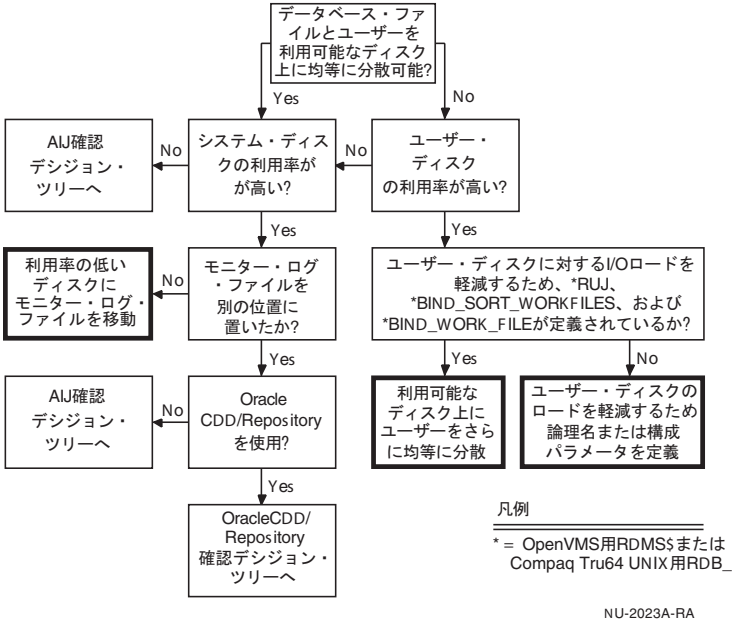
ディスク・デバイスに問題があると考えられない場合、I/O リソース・ボトルネックに対するチェックを行う別の領域にアプリケーションがあります。アプリケーションのダイレクト I/O レートが平均的に高ければ、リクエストをサービスするディスクによって影響を受ける可能性があります。ディスク・リクエストは、メモリー・リクエストに比べて、比較的遅いので、アプリケーションのダイレクト I/O レートが高ければ、チューニングが可能な領域です。

丁寧に調査した結果 I/O リソースには問題がないことがわかったら、次に検査を行う領域はメモリーです。8.2 項（8-38）で、メモリー・リソースの調査方法について説明します。

8.1.2 I/O ロードの均衡化

マルチファイル・データベースであれば、データベース・ホット・スポットを削減するように、利用できるディスクに渡ってデータベースが均等に分散されていることを確認します。データベース・ユーザーのデフォルト・ディレクトリがユーザー・ディスクに渡って均等に分散されていれば、同じように競合問題を防ぐ支援になります。図 8-2（8-4）では、I/O ロードの均衡化のステップを要約します。システム・ディスクの利用率が高ければ、競合を低減するためデフォルトの Oracle Rdb モニター・ログ・ファイルの場所を別のディスクに移動します。

図 8-2 デシジョン・ツリー： I/O ロードの均衡化



ユーザーのディスクの利用率が高ければ、論理名 RDMS$BIND_SORT_WORKFILES、RDMS$RUJ および RDMS$BIND_WORK_FILE または構成パラメータ RDB_BIND_SORT_WORKFILES、RDB_RUJ および RDB_BIND_WORK_FILE を定義すると、競合を低減できます。これらの論理名と構成パラメータを使用すると、Oracle Rdb 一時ファイルのデフォルトの位置を変えることができます。

OpenVMS VAX OpenVMS Alpha

OpenVMS では、RDMS$BIND_SORT_WORKFILES 論理名は OpenVMS Sort ユーティリティ（SORT）で使用される一時ファイルの数を定義しています。Oracle Rdb は、たとえば、大きな SORTED BY クエリーおよび大きなインデックスの構築など、オプティマイザがメモリー内でソートを行うには大きすぎると決定したすべてのソート操作に対して SORT を使用します。RDMS$BIND_SORT_WORKFILES の最大値は、10（ファイル）です。デフォルト値は、2 です。

RDMS$BIND_SORT_WORKFILES はソート作業ファイルに使用するファイルの数を制御しますが、SORTWORK0 から SORTWORK9 はソート作業ファイルの置き場所を制御します。ソート操作を行う間の競合を低減するため、これらのファイルはいくつかのディスク上に分散させます。◆

また、RDMS$BIND_SORT_WORKFILES または RDB_BIND_SORT_WORKFILES と関連するソート作業ファイルを各アプリケーションに対して詳細に定義できます。詳細は、A.89 項（A-42）を参照してください。

RDMS$RUJ 論理名と RDB_RUJ 構成パラメータは、.ruj ファイルのデフォルトの場所を制御します。この名前を他の方法で定義すると、ユーザー・ディスク上の競合を削減できます。次の例は、OpenVMS で .ruj ファイルをリダイレクトする方法を示しています。

```
$ DEFINE RDMS$RUJ RUJ$DISK:[OFFLOADED_RUJS]
```

最後に、RDMS$BIND_WORK_FILE 論理名または RDB_BIND_WORK_FILE 構成パラメータを使用すると、Oracle Rdb が作成した一時ファイルを別の場所にリダイレクトすることができます。また、この論理名または構成パラメータを使用して、ユーザー・ディスク上の競合を削減できます。次の例では、OpenVMS 上で作成されるこれらの一時ファイルの置き場所をリダイレクトする方法を示します。

```
$ DEFINE RDMS$BIND_WORK_FILE WORK$DISK:[OFFLOADED_TEMP_FILES]
```

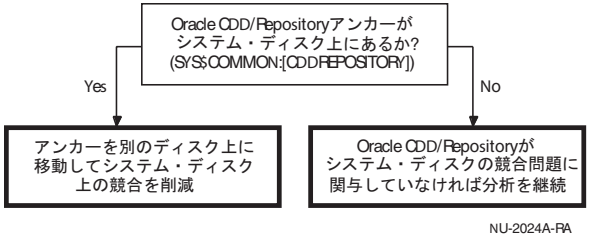
システム・ディスクからすべてのデータベース・ファイルやリポジトリ・ファイルを他のデバイスに移動した後でも、システム・ディスクにボトルネックが残っているならば、高いページ・ファイル・アクティビティがボトルネックの原因である可能性があります。ハード・ページ・フォルト率が高く、シングル・ページ・ファイルのみが使用されている証拠がある場合、他のデバイスに 2 つめのページ・ファイルを追加すれば、ボトルネックが軽減されるでしょう。この状況は、プロセスの仮想メモリー要件が作業セットの割当てよりもきわめて大きいようなデータベース環境でときどき発生します。データベース・バッファ・パラメータは、プロセス上の仮想メモリーの要件に最大の影響を与えます。

8.1.2.1 Oracle CDD/Repository の確認

OpenVMS VAX OpenVMS Alpha

Oracle CDD/Repository が広範囲に渡って使用されていて、最後のディレクトリが置かれているディスクの利用率が高い場合、そのディスク上の競合を低減するオプションの 1 つは、CDD アンカーの位置を他のディスクに移動させることです。図 8-3（8-6）を参照してください。

図 8-3 デシジョン・ツリー： Oracle CDD/Repository の確認

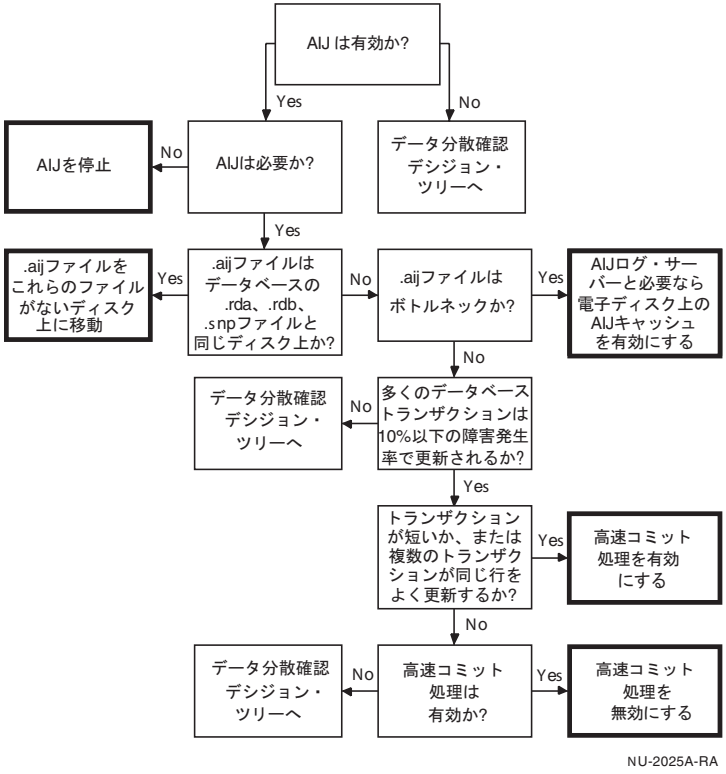


VMScluster 環境での Oracle CDD/Repository の使用の説明は、「6.2.7 項 Oracle CDD/Repository 要件」（6-19）を参照してください。◆

8.1.2.2 AIJ の確認

After-image ジャーナルが有効であって、.aij ファイルが書き込まれるディスクの利用率が高い場合、チューニングの可能性の 1 つに、この .aij ファイルを利用率が低いディスクへ移動させるという方法があります（図 8-4（8-7）参照）。.aij ファイルの場所は、論理名または構成パラメータを使用しても変更できません。そのかわり、.aij ファイルを移動するには、SQL の ALTER DATABASE 文を使用します。JOURNAL ALLOCATION IS と JOURNAL EXTENT IS 句を使用すると、AIJ が書き込むパフォーマンスをさらに拡張できます。これらの句は、ジャーナル・ファイルの最初の割当てが十分であることを保証するので、トランザクションはエクステントが発生するまで待つ必要がありません。

図 8-4 デシジョン・ツリー： AIJ の確認



.aij ファイルを独自のディスクかデータベース以外のディスクに移動する別の理由は、このようにするとデータベースの信頼性が高まるためです。.aij ファイルを使用すると、.rda、.rdb または .snp ファイルが存在するディスクの 1 つで障害が発生した場合、データベース管理者（DBA）が前回のバックアップ場所からデータベースをロールフォワードできます。このため、.aij ファイルは、データベースの .rda、.rdb および .snp ファイルとは別のディスクに置く必要があります。データベースに対して複数の .aij ファイルを使用している場合、各 .aij ファイルはデータベースの .rda、.rdb および .snp ファイルとは別のディスク上に置く必要があります。そうしなければ、ディスク障害が発生した場合、回復できない可能性があります。

常に After-image ジャーナルを有効にしておくのは優れた方法です。Oracle Rdb は、データベースが読取り専用である場合や障害が発生したときに回復する必要がない場合を除いて、AIJ ジャーナルを有効にしておくようお薦めします。

環境によっては、高速コミット処理を有効にすることで、データベースのパフォーマンスを改善することができます。高速コミットを有効にする場合、Oracle Rdb はトランザクションがコミットされたときに更新したページをバッファ・プールに保存し、ディスクにはページを書き込みません。更新されたページは、非同期バッチ書込み操作の一部としてデータベースに書き込まれるまで、バッファ・プール内に残ります。ユーザーが指定したしきい値（チェックポイント）に達すると、複数のトランザクションに対して更新されたページがすべてディスクに書き込まれます。トランザクションに障害が発生すると、更新されたページがディスク上に書き込まれていないので、以前のすべてのトランザクションは前回のチェックポイントまで戻ってやり直す必要があります。Oracle Rdb は、.aij ファイルに書き込まれた情報を使用して、これらのトランザクションを再実行します。

安定したトランザクション処理環境があり、プロセスが複数のトランザクションに対して同じ行を更新するか、トランザクションが短くて、多くのページを更新しないという条件が存在する場合、高速コミット・トランザクション処理を有効することを考慮する必要があります。一般的に、これらのタイプのトランザクションは、バッファ・プール・オーバフローを起こしにくい傾向があります。これらのタイプのトランザクションではデータベースへの I/O は削減されます。つまり、トランザクションで障害が発生して、高速コミット・トランザクション処理が有効である場合に行われる再実行操作は、多くのページを更新する長いトランザクションに対する再実行操作に比べると短いことを意味しています。高速コミット・トランザクション処理の情報は、4.1.5 項（4-89）を参照してください。

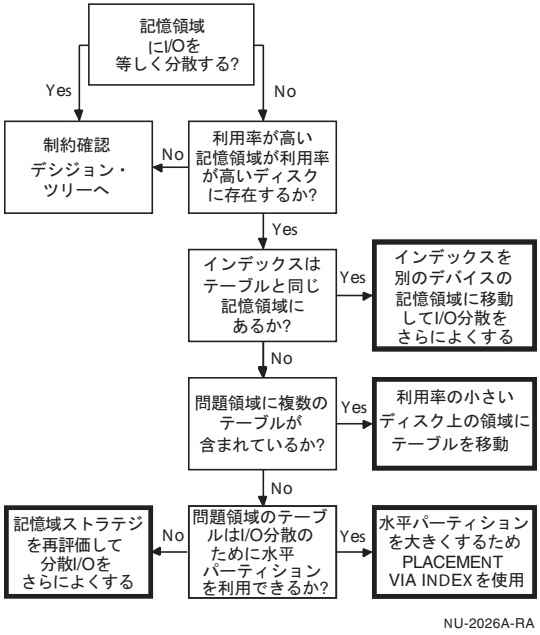
高速コミット・トランザクション処理を有効にした場合、ジャーナル最適化も有効にできます。このオプションを使用すると、データベース・ルートに対する大部分の I/O が取り除かれるので、コミット処理速度が非常に速くなります。このオプションは、更新集中型のデータベース環境のパフォーマンスを拡張します。

高速コミット処理と AIJ 最適化の詳細は、4.1.5 項（4-89）を参照してください。

8.1.2.3 データ分散の確認

図 8-5（8-9）は、データ分散を確認するためにのステップを要約します。

図 8-5 デシジョン・ツリー： データ分散の確認



データベース記憶領域に渡って I/O が分散していることを測定する 1 つの方法は、Performance Monitor で「IO Statistics」画面を使用することです。これは、I/O アクティビティを与えられた時間内に記憶領域とディスクに均等に分散させる方法についての一般的なアイデアを与えます。これを測定する 1 つの方法は、一定時間 Performance Monitor を実行して、「IO Statistics (by file)」オプションを入力し、「Option」メニューを表示して、レポートを記述します。これは、その期間に対するすべてのメニューの ASCII 出力を含むファイル（STATISTICS.RPT）を生成します。

好みのエディタを使用して、このファイルを編集できます。また、任意のプリンタを使用してプリントできます。STATISTICS.RPT ファイルには、例 8-3（8-10）で示したセクションと同様のセクションがあります。

例 8-3 「IO Statistics (By File)」画面

```
Node: MYNODE          Oracle Rdb V7.0-00 Performance Monitor 30-MAY-1996 14:24:45
Rate: 3.00 Seconds              File IO Statistics            Elapsed: 03:04:56.88
Page: 1 of 1      USER18:[PRODUCTION.DATABASE]PRODUCT_DB.RDB;1       Mode: Online
--------------------------------------------------------------------------------
                          For File: All data/snap files
statistic..... rate.per.second.............. total....... average......
name.............. max..... cur..... avg....... count....... per.trans....
total I/Os                          24                  39152          76.5
    (Synch. reads)          48       10       22.0       35254          68.9
    (Synch. writes)         16       14        2.4        3898           7.6
    (Extends)                0        0        0.0           0           0.0
    (Asynch. reads)          0        0        0.0           0           0.0
    (Asynch. writes)         0        0        0.0           0           0.0
statistic..... blocks.transferred.........  stall.time.(x100)...........
name.............. avg.per.I/O.. total.........  avg.per.I/O... total........
total I/Os                   5.3        209348             3.5        137914
    (Synch. reads)           5.7        199280             3.8        133681
    (Synch. writes)          2.6         10068             1.1          4233
    (Extends)                0.0             0             0.0             0
    (Asynch. reads)          0.0             0             0.0             0
    (Asynch. writes)         0.0             0             0.0             0
--------------------------------------------------------------------------------
Exit Help Menu Options Reset Set_rate Unreset Write !
```

I/O の総計セクション、すなわち 1 秒あたりの最大と平均、総カウント値は、各領域に送られる I/O の割合の決定に使用できます。Total I/O の数値が小さければ、これ以上最適化しても、重要でない結果しか得られません。4.2.1.4 項（4-137）は、「IO Statistics (by file)」画面の情報も提供します。

停止時間が長ければ、アプリケーションの応答時間が長くなるので、「IO Statistics」画面上の停止時間の値を見る必要があります。すべてのデータ・ファイルで「IO Statistics」画面（例 8-3（8-10））を使用して統計を表示するときに、データベース内の .rda および .snp ファイルの個々に対して I/O ごとの停止時間の平均を表示した場合、"stall time avg per I/O" の値が期待した値とは違っている可能性があります。たとえば、すべてのデータ・ファイル画面に対する "stall time avg per I/O" フィールドの値が、個々の .rda および .snp ファイルに対する "stall time avg per I/O" フィールドの値の平均よりも小さい可能性があります。Oracle Rdb は、書込み I/O 操作を並行、非同期に発行します（バッチ書込みメカニズム）。これは、すべてのデータベース・ファイル画面に対する I/O あたりの平均停止時間は、個々の .rda および .snp ファイルに対して表示された "average of the averages" ではないこと意味します。これは、I/O は連続して完了することを暗示しています。むしろ、総 I/O 停止時間は、"the average of the averages divided by the number of I/Os" であり、停止時間は並行して発行されるすべての I/O に渡って償却されるので、すべてのデータ・ファイル画面に対する平均は、通常個々のファイルの平均の一部になります。

たとえば、Oracle Rdb が 4 つの記憶領域でバッチ書込み操作を行っているとします（もちろん、すべて並列）。個々の記憶領域の I/O 操作は、それぞれ 20 ミリ秒かかるとします。I/O が連続して行われているとすると、すべての記憶領域の平均停止時間は 20 ミリ秒になります。しかし、I/O は並列に行われているので、すべての領域の平均は実際 5 ミリ秒です（20 ミリ秒を 4 回の I/O 操作で割ります）。

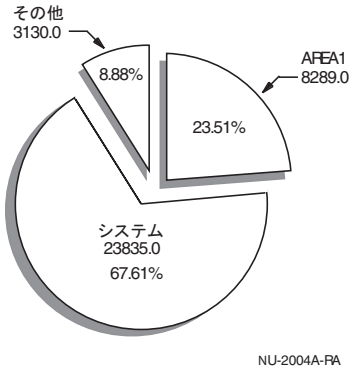
非同期のバッチ書込み操作が有効であれば、Oracle Rdb はデータベースに書込み操作を実行し、そして、書込み操作が完了するまで待つ必要はありません。非同期書込みの I/O は「IO Statistics (by file)」画面に記録されますが、「IO Statistics (by file)」画面には、これらの書込みに対する停止時間はほとんどの非同期バッチ書込み操作から除かれているため記録されません。非同期バッチ書込み操作の説明は、3.2.5 項（3-25）を参照してください。

ディスクの I/O 操作に 30 ミリ秒かかるとするならば、「IO Statistics (by file)」画面上の I/O あたりの平均停止時間が 30 ミリ秒を超えた場合のみ停止時間を改善（削減）しようと考えます。I/O 操作に対する平均停止時間が 30 ミリ秒よりも小さい場合、これは、先ほど説明したようにバッチ書込み操作によって実行された並列書込みによって得られたものです。ディスク・パフォーマンス統計を検査して、ディスク・パフォーマンスの変更を監視できます。

図 8-6（8-12）は、各記憶領域によって生成された領域総読取り I/O の割合を示しています。これは、各領域における読取り I/O の総数を総読取り I/O で割って、その結果に 100 をかけたものです。次の表は、すべてのデータベース・ファイルの総読取り I/O を示しています（例 8-3（8-10）を参照）。2 つの記憶領域（SYSTEM_DB および AREA1）は、例 8-4（8-12）および例 8-6（8-13）にそれぞれ現れており、.rdb、.snp、.ruj および .aij ファイルを含むその他のファイルとともに表示されています。その他のファイル I/O は、すべての記憶領域の読取り I/O の総数を総数から引いたものです。

	読取り I/O	総読取り I/O の割合
総計	35254	
システム領域	23835	67.61
Area1	8289	23.51
その他	3130	8.88

図 8-6 記憶領域ごとの読取り I/O の割合



すべての例（例 8-3（8-10）から例 8-7（8-14））から得た総停止時間列は、読取り I/O または書込み I/O またはその両方に関連していることを判断する場合に便利です。停止時間が長くなれば、アプリケーションからの応答時間が遅くなります。データベースに対する非同期プリフェッチ機能を有効にすると、3.2.4 項（3-23）で説明したように順次読取り操作に対する停止時間をなくすことができます。

例 8-4（8-12）から例 8-7（8-14）からまでの報告では、最大の I/O を生成しているこれらの記憶領域に対するデータとスナップショット・ファイル I/O 測定を示します。システム領域が 67% 以上の読取り I/O を生成しており、最大値が高いので、現行の記憶域の構造では、ボトルネックが発生する可能性があります。ビューを使用してディスクの利用率に対してさらに調査を行ったところ、ディスク間に渡る I/O に対してさらにバランスを取る必要があります。例 8-4（8-12）は、システム記憶領域に対する I/O 統計を表示しています。

例 8-4 システム記憶領域に対する I/O 統計

```
Node: MYNODE          Oracle Rdb V7.0-00 Performance Monitor 30-MAY-1996 14:29:37
Rate: 3.00 Seconds              File IO Statistics           Elapsed: 00:00:12.17
Page: 1 of 1      USER18:[PRODUCTION.DATABASE]PRODUCT_DB.RDB;1      Mode: Online
-------------------------------------------------------------------------------
            For File: USER07:[PRODUCTION.DATABASE.SYSTEM]SYSTEM_DB.RDA;1
statistic..... rate.per.second..... total..... average.....
name.............. max..... cur..... avg..... count..... per.trans....
total I/Os                          10                  25078          49.0
    (Synch. reads)          48       4       14.9       23835          46.6
    (Synch. writes)          6       6        0.8        1243           2.4
    (Extends)                0       0        0.0           0           0.0
    (Asynch. reads)          0       0        0.0           0           0.0
    (Asynch. writes)         0       0        0.0           0           0.0
statistic..... blocks.transferred.........  stall.time.(x100).....
```

```
name.............. avg.per.I/O.. total........  avg.per.I/O... total........
total I/Os                  5.5        138208             4.6        115420
    (Synch. reads)          5.7        135490             4.4        105137
    (Synch. writes)         2.2          2718             8.3         10283
    (Extends)               0.0             0             0.0             0
    (Asynch. reads)         0.0             0             0.0             0
    (Asynch. writes)        0.0             0             0.0             0
--------------------------------------------------------------------------------
Exit Help Menu Options Reset Set_rate Unreset Write !
```

例 8-5（8-13）は、システム・スナップショット・ファイルに対する I/O 統計を示しています。

例 8-5 システム・スナップショット・ファイルに対する I/O 統計

```
Node: MYNODE          Oracle Rdb V7.0-00 Performance Monitor 30-MAY-1996 14:35:21
Rate: 3.00 Seconds             File IO Statistics           Elapsed: 00:05:55.96
Page: 1 of 1     USER18:[PRODUCTION.DATABASE]PRODUCT_DB.RDB;1       Mode: Online
--------------------------------------------------------------------------------
            For File: USER19:[PRODUCTION.DATABASE.SYSTEM]SYSTEM_DB.SNP;1
statistic........... rate.per.second.............. total....... average......
name.............. max..... cur..... avg....... count....... per.trans....
total I/Os                          4                     1215           2.4
    (Synch. reads)          2       2       0.3          547           1.1
    (Synch. writes)         3       2       0.4          668           1.3
    (Extends)               0       0       0.0            0           0.0
    (Asynch. reads)         0       0       0.0            0           0.0
    (Asynch. writes)        0       0       0.0            0           0.0
statistic........... blocks.transferred.........  stall.time.(x100)...........
name.............. avg.per.I/O.. total........  avg.per.I/O... total........
total I/Os                  4.9          5914             4.6          5631
    (Synch. reads)          6.0          3266             3.3          1809
    (Synch. writes)         4.0          2648             5.7          3822
    (Extends)               0.0             0             0.0             0
    (Asynch. reads)         0.0             0             0.0             0
    (Asynch. writes)        0.0             0             0.0             0
--------------------------------------------------------------------------------
Exit Help Menu Options Reset Set_rate Unreset Write !
```

例 8-6（8-13）は、AREA1 記憶領域に対する I/O 統計を示しています。

例 8-6 AREA1 記憶領域に対する I/O 統計

```
Node: MYNODE          Oracle Rdb V7.0-00 Performance Monitor 30-MAY-1996 14:37:55
Rate: 3.00 Seconds             File IO Statistics           Elapsed: 00:08:29.96
Page: 1 of 1      USER18:[PRODUCTION.DATABASE]PRODUCT_DB.RDB;1       Mode: Online
--------------------------------------------------------------------------------
            For File: USER16:[PRODUCTION.DATABASE.AREA1]AREA1.RDA;1
statistic........... rate.per.second.............. total....... average......
name.............. max..... cur..... avg....... count....... per.trans....
```

```
total I/Os                          7                  9313          18.2
    (Synch. reads)         39       3       5.2        8289          16.2
    (Synch. writes)         5       4       0.6        1024           2.0
    (Extends)               0       0       0.0           0           0.0
    (Asynch. reads)         0       0       0.0           0           0.0
    (Asynch. writes)        0       0       0.0           0           0.0
statistic..... blocks.transferred.........  stall.time.(x100)...........
name.............. avg.per.I/O.. total.....  avg.per.I/O... total.....
total I/Os                 5.3        49702          2.7         25104
    (Synch. reads)         5.7        47654          2.2         18545
    (Synch. writes)        2.0         2048          6.4          6559
    (Extends)              0.0            0          0.0             0
    (Asynch. reads)        0.0            0          0.0             0
    (Asynch. writes)       0.0            0          0.0             0
--------------------------------------------------------------------------------
Exit Help Menu Options Reset Set_rate Unreset Write !
```

例 8-7（8-14）は、AREA1 スナップショット・ファイルに対する I/O 統計を表示しています。

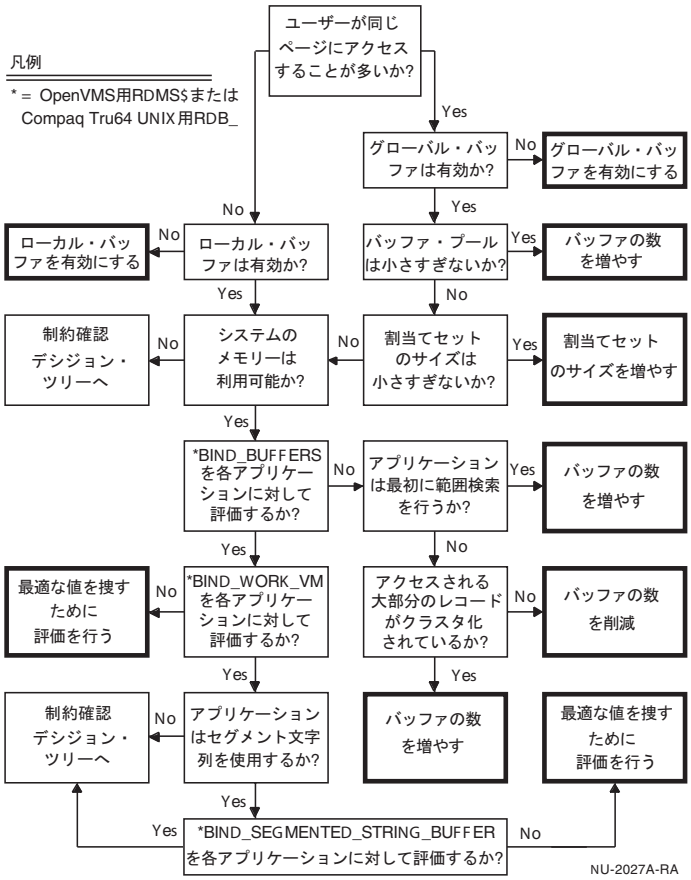
例 8-7 AREA1 スナップショット・ファイルに対する I/O 統計

```
Node: MYNODE         Oracle Rdb V7.0-00 Performance Monitor 30-MAY-1996 14:41:26
Rate: 3.00 Seconds             File IO Statistics          Elapsed: 00:12:00.67
Page: 1 of 1      USER18:[PRODUCTION.DATABASE]PRODUCT_DB.RDB;1      Mode: Online
--------------------------------------------------------------------------------
            For File: USER29:[PRODUCTION.DATABASE.AREA1]AREA1.SNP;1
statistic..... rate.per.second.............. total..... average......
name.............. max..... cur..... avg..... count..... per.trans....
total I/Os                          2                   797           1.6
    (Synch. reads)          1       1       0.2         254           0.5
    (Synch. writes)         3       1       0.3         543           1.1
    (Extends)               0       0       0.0           0           0.0
    (Asynch. reads)         0       0       0.0           0           0.0
    (Asynch. writes)        0       0       0.0           0           0.0
statistic..... blocks.transferred.........  stall.time.(x100)...........
name.............. avg.per.I/O.. total.....  avg.per.I/O... total.....
total I/Os                 4.1         3306          5.9          4701
    (Synch. reads)         6.0         1524          4.0          1005
    (Synch. writes)        3.3         1782          6.8          3696
    (Extends)              0.0            0          0.0             0
    (Asynch. reads)        0.0            0          0.0             0
    (Asynch. writes)       0.0            0          0.0             0
--------------------------------------------------------------------------------
Exit Help Menu Options Reset Set_rate Unreset Write !
```

8.1.3 I/O 操作の削減

ダイレクト I/O 操作の数を削減する方法の 1 つは、この操作に対する必要性をなくすことです。これは、システム、データベースまたはアプリケーション・レベルの変更で行えます。図 8-7（8-15）を参照してください。

図 8-7 デシジョン・ツリー： I/O の削減



ディスク I/O を制御する方法の 1 つに、バッファ管理を通した方法があります。Oracle Rdb を使用すると、ローカル・バッファまたはグローバル・バッファを使用できます。バッファ管理とローカル・バッファおよびグローバル・バッファの相対的利点の一般的な説明は、4.1.2 項（4-19）を参照してください。この項の残りの部分では、ローカル・バッファとグローバル・バッファの使用とその調整方法を説明します。

システム・レベルから I/O を削減するために最もよく使用される方法は、バッファ・プール・サイズを大きくする方法で、ディスク I/O を行うかわりに増加したメモリーを参照します。このためには、システム上に追加メモリー容量が必要です。バッファへのアクセスは、ディスクへのアクセスと比べて非常に高速なので、特に遅いディスクや利用率が高いディスク・デバイスへアクセスしている場合、この方法を使用すると応答時間を十分に改善できます。

I/O バッファ・プール・サイズを制御する方法の 1 つに、データベースが維持しているバッファ数を変更する方法があります。これは、データベース内で SQL の ALTER DATABASE 文の NUMBER OF BUFFERS 句を使用するか、システム・レベルで RDM$BIND_BUFFERS 論理名か、RDB_BIND_BUFFERS 構成パラメータを通して行うことができます。このパラメータは、動的であるので、非常に少ないリスクまたはコストで変更できます。データベース・バッファのデフォルト数は、20 です。

OpenVMS VAX OpenVMS Alpha

論理名は、バッファを調整するため、システム、グループまたはプロセスに渡って設定できます。◆

NUMBER OF BUFFERS 句の詳細情報は、4.1.2.3 項（4-27）を参照してください。

データベース・バッファの数を増やしたことによって必要な追加メモリー量の決定を支援するため、バッファの数を 20 ずつ増やしながらデータベース・アプリケーションを実行し、メモリー統計を収集します。図 8-8（8-17）は、この実験の結果を示しています。バッファ・サイズは一定のままなので、バッファの数を増やすと、予測した通り、仮想メモリーの消費に線形に影響します。バッファ・サイズを変更すると、どちらの方向にバッファ・サイズを変えるかによって、一定の割合で線が上方または下方に移動します。しかし、調査したアプリケーションの性質によって、結果は劇的に異なる可能性があるので、同じような実験を自分のデータベースで実行して、バッファ数が増加したために追加する消費メモリーの総計を決定する必要があります。

図 8-8 仮想メモリーの消費量とバッファの数

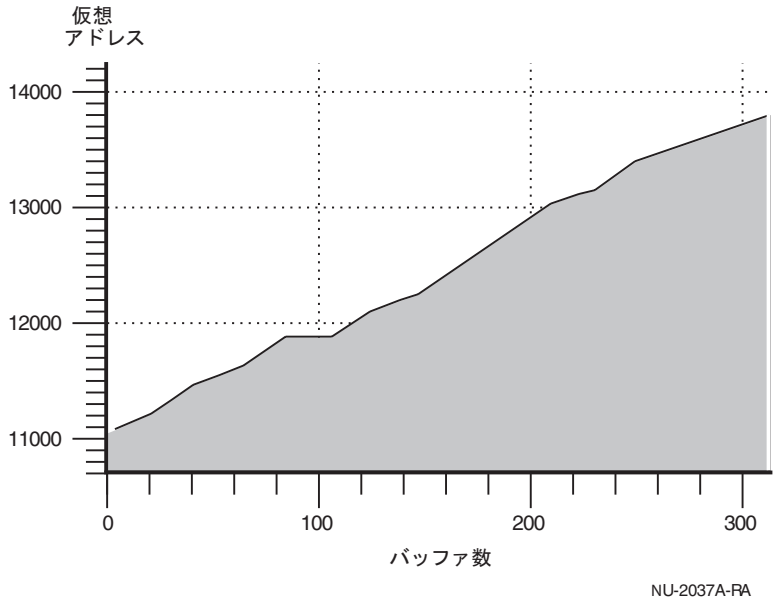
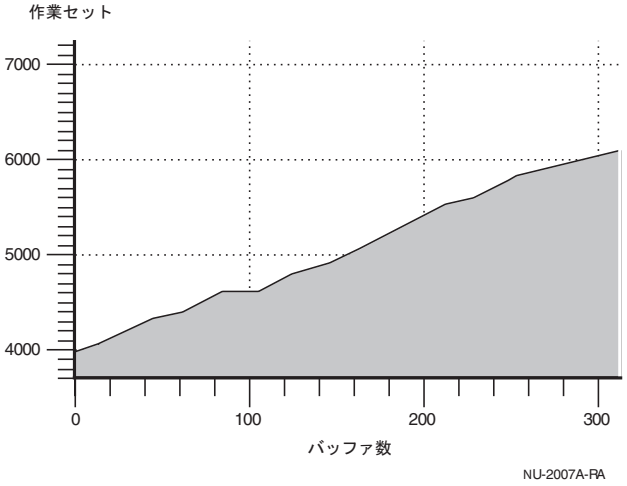


図 8-9（8-18）は、データベース・バッファの数が増加することで、作業セットのサイズがどのように影響するかを同じアプリケーションで 20 バッファずつ増やした場合で示しています。結果は、作業セットの消費は同じようにほとんど線形になっていることを示しています。

図 8-9 作業セットのサイズとバッファの数



これは、応答時間に対して何を意味するのでしょうか。応答時間は、ディスク速度やディスクの利用率、アプリケーションが実行しているトランザクションのタイプ、更新が発生する頻度およびデータベース内の競合の量によって変わります。一般的な規則として、範囲検索を行っているアプリケーションはバッファが多いほどその恩恵を受けます。データがクラスタ化されていれば、最初の情報の一部にアクセスしたとき関連する情報がバッファに読み込まれることも確かです。ロード集中型の作業負荷は、バッファ数が少ないほど利点が得られます。ロード集中型作業負荷は主に物理データベース・ページを取り扱うので、大量のバッファを維持するオーバーヘッドを負うことに大きな意味はありません。

応答時間は、データベース・バッファの数に対して線形ではありません。

バッファ・プールをデータベースに渡って効果的に測定する 1 つの確実な方法は、Performance Monitor の「PIO Statistics--Data Fetches」画面および「PIO Statistics--SPAM Fetches」画面を通して行います。これらの画面のローカル・バッファ版を、例 8-8（8-19）に示します。これらの画面のグローバル・バッファ版は、例 8-9（8-20）に示します。

データベースのバッファ・プールの効果を決定する方法は、データベースがローカル・バッファを使用するか、グローバル・バッファを使用するかによって決まります。

データベースの持つローカル・バッファが有効になっている場合、例 8-8（8-19）に示すようにローカル・バッファ版の「PIO Statistics--Data Fetches」画面と「PIO Statistics--SPAM Fetches」画面が Performance Monitor によって表示されます。この画面の詳細は、Performance Monitor のヘルプを参照してください。

例 8-8 Performance Monitor のローカル・バッファ版の「PIO Statistics--Data Fetches」画面と「PIO Statistics--SPAM Fetches」画面

```
Node: MYNODE         Oracle Rdb V7.0-00 Performance Monitor 30-MAY-1996 14:48:04
Rate: 3.00 Seconds        PIO Statistics--Data Fetches      Elapsed: 00:18:38.25
Page: 1 of 1       RDBVMS_USER1:[LOGAN.V70]MF_PERSONNEL.RDB;1       Mode: Online
-------------------------------------------------------------------------------
statistic.........      rate.per.second.............. total....... average......
name..............      max..... cur..... avg....... count....... per.trans....
fetch for read              2287        0       20.4        18524         926.2
fetch for write               58        0        0.4          348          17.4
in LB: all ok               2035        0       18.3        16631         831.6
   LB: need lock             296        0        2.1         1949          97.5
   LB: old version             0        0        0.0            0           0.0
not found: read               34        0        0.3          292          14.6
         : synth               0        0        0.0            0           0.0
DAPF: success                  0        0        0.0            0           0.0
DAPF: failure                  0        0        0.0            0           0.0
DAPF: utilized                 0        0        0.0            0           0.0
DAPF: discarded                0        0        0.0            0           0.0
-------------------------------------------------------------------------------
Exit Graph Help Menu Options Pause Reset Set_rate Time_plot Unreset Write X_plot
Node: MYNODE         Oracle Rdb V7.0-00 Performance Monitor 30-MAY-1996 14:51:28
Rate: 3.00 Seconds        PIO Statistics--SPAM Fetches      Elapsed: 00:22:03.13
Page: 1 of 1       RDBVMS_USER1:[LOGAN.V70]MF_PERSONNEL.RDB;1       Mode: Online
-------------------------------------------------------------------------------
statistic.........      rate.per.second.............. total....... average......
name..............      max..... cur..... avg....... count....... per.trans....
fetch for read                 4        0        0.1           59           3.0
fetch for write                0        0        0.0            0           0.0
in LB: all ok                  3        0        0.1           52           2.6
   LB: need lock               1        0        0.0            6           0.3
   LB: old version             0        0        0.0            0           0.0
not found: read                0        0        0.0            1           0.1
         : synth               0        0        0.0            0           0.0
DAPF: success                  0        0        0.0            0           0.0
DAPF: failure                  0        0        0.0            0           0.0
DAPF: utilized                 0        0        0.0            0           0.0
DAPF: discarded                0        0        0.0            0           0.0
-------------------------------------------------------------------------------
Exit Graph Help Menu Options Pause Reset Set_rate Time_plot Unreset Write X_plot
```

ローカル・バッファが有効になっているデータベースの場合、次に示すようにローカル・バッファ・プール内で見つかったページに対して要求されたページの割合を計算できます。

```
( total number found in local buffer pool  / total number of page requests ) × 100
```

"total number found in local buffer pool" カウントを計算するには、「PIO Statistics--Data Fetches」画面上の "in LB: all ok" と "LB: need lock" カウントの合計を「PIO Statistics--SPAM Fetches」画面上の "in LB: all ok" と "LB: need lock" カウントの合計に加えます。"total number of page requests" カウントを計算するには、「PIO Statistics--Data Fetches」画面上の "fetch for read" と "fetch for write" のカウントの合計を「PIO Statistics--SPAM Fetches」画面上の "fetch for read" と "fetch for write" のカウントの合計に加えます。

例 8-8（8-19）では、バッファ・プールの有効性は次のようになります。

```
(18638 / 18931) × 100 = 約 98.5%
```

一般的に、割合が高くなるにつれて、バッファリングがよくなります。すべてのページ要求に対して各ページがバッファ・プール内で見つかれば、バッファ・プールの有効性は 100% になります。

データベースの持つグローバル・バッファが有効になっている場合、例 8-9（8-20）に示すようにグローバル・バッファ版の「PIO Statistics--Data Fetches」画面および「PIO Statistics--SPAM Fetches」画面が Performance Monitor によって表示されます。

例 8-9 Performance Monitor のグローバル・バッファ版の「PIO Statistics--Data Fetches」画面と「PIO Statistics--SPAM Fetches」画面

```
Node: MYNODE         Oracle Rdb V7.0-00 Performance Monitor 30-MAY-1996 15:47:52
Rate: 3.00 Seconds        PIO Statistics--Data Fetches      Elapsed: 00:00:46.03
Page: 1 of 1       RDBVMS_USER1:[LOGAN.V70]MF_PERSONNEL.RDB;1       Mode: Online
-------------------------------------------------------------------------------
statistic.........      rate.per.second..... total....... average......
name..............      max..... cur..... avg....... count....... per.trans....
fetch for read              1872        0       16.2        18524         926.2
fetch for write               39        0        0.3          348          17.4
in AS: all ok               1693        0       14.1        16101         805.1
   AS: lock for GB             0        0        0.0            0           0.0
   AS: need lock             127        0        1.3         1494          74.7
   AS: old version             0        0        0.0            0           0.0
in GB: need lock              73        0        1.0         1107          55.4
   GB: old version             0        0        0.0            0           0.0
   GB: transferred             0        0        0.0            0           0.0
not found: read               18        0        0.1          170           8.5
         : synth               0        0        0.0            0           0.0
DAPF: success                  0        0        0.0            0           0.0
DAPF: failure                  0        0        0.0            0           0.0
DAPF: utilized                 0        0        0.0            0           0.0
DAPF: discarded                0        0        0.0            0           0.0
-------------------------------------------------------------------------------
Exit Graph Help Menu Options Pause Reset Set_rate Time_plot Unreset Write X_plot
Node: MYNODE         Oracle Rdb V7.0-00 Performance Monitor 30-MAY-1996 15:51:47
Rate: 3.00 Seconds        PIO Statistics--SPAM Fetches      Elapsed: 00:04:41.06
Page: 1 of 1       RDBVMS_USER1:[LOGAN.V70]MF_PERSONNEL.RDB;1       Mode: Online
```

```
-------------------------------------------------------------------------------
statistic.........      rate.per.second.............. total....... average......
name..............      max..... cur..... avg....... count....... per.trans....
fetch for read                5        0        0.1          59           3.0
fetch for write               0        0        0.0           0           0.0
in AS: all ok                 4        0        0.0          52           2.6
   AS: lock for GB            0        0        0.0           0           0.0
   AS: need lock              0        0        0.0           0           0.0
   AS: old version            0        0        0.0           0           0.0
in GB: need lock              1        0        0.0           6           0.3
   GB: old version            0        0        0.0           0           0.0
   GB: transferred            0        0        0.0           0           0.0
not found: read               0        0        0.0           1           0.1
         : synth              0        0        0.0           0           0.0
DAPF: success                 0        0        0.0           0           0.0
DAPF: failure                 0        0        0.0           0           0.0
DAPF: utilized                0        0        0.0           0           0.0
DAPF: discarded               0        0        0.0           0           0.0
-------------------------------------------------------------------------------
Exit Graph Help Menu Options Pause Reset Set_rate Time_plot Unreset Write X_plot
```

グローバル・バッファが有効になっているデータベースの場合、次が計算できます。

- 要求したページが割当てセット内で見つかった割合
- 要求したページがグローバル・バッファ・プール内で見つかった割合
- データベースのローカル・バッファが有効であれば起こるはずだったディスク I/O をページ要求の何パーセント節約できたか

割当てセット内でページを見つけたページ要求の割合は、次の式で計算できます。

```
( total number found in allocate set  / total number of page requests ) × 100
```

"total number found in allocate set" カウントを計算するには、「PIO Statistics--Data Fetches」画面上の "in AS: all ok"、"AS: lock for GB" と "AS: need lock" カウントの合計を「PIO Statistics--SPAM Fetches」画面上の "in AS: all ok"、"AS: lock for GB" と "AS: need lock" カウントの合計に加えます。"total number of page requests" カウントを計算するには、「PIO Statistics--Data Fetches」画面上の "fetch for read" と "fetch for write" のカウントの合計を「PIO Statistics--SPAM Fetches」画面上の "fetch for read" と "fetch for write" のカウントの合計に加えます。

例 8-9（8-20）では、割当てセット内で見つかったページに対するページ要求の割合は、次のように計算できます。

```
(17647 / 18931) × 100 = 約 93.2%
```

ページ要求に対してグローバル・バッファ・プール（これには、割当てセット内で見つかったページも含まれます）内で見つかったページの割合を次のように計算できます。

```
( total number found in global buffer pool / total number of page requests ) × 100
```

"total number found in global buffer pool" カウントを計算するには、「PIO Statistics--Data Fetches」画面上の "in AS: all ok"、"AS: lock for GB"、"AS: need lock" と "In GB: need lock" カウントの合計を「PIO Statistics--SPAM Fetches」画面上の "in AS: all ok"、"AS: lock for GB"、"AS: need lock" と "In GB: need lock" カウントの合計に加えます。"total number of page requests" カウントを計算するには、「PIO Statistics--Data Fetches」画面上の "fetch for read" と "fetch for write" のカウントの合計を「PIO Statistics--SPAM Fetches」画面上の "fetch for read" と "fetch for write" のカウントの合計に加えます。

例 8-9（8-20）では、グローバル・バッファ・プール内（割当てセットを含む）で見つかったページに対するページ要求の割合は、次のように計算できます。

```
(18760 / 18931) × 100 = 約 99.1%
```

割当てセットの効率とバッファ・プールの効率を計算した後、各プロセス（割当てセット）に対して割り当てたグローバル・バッファの数とバッファ・プール内のバッファの数が適切であるかを判断できます。データベースに対するバッファ・プールの効率を上げようとして、システム上に追加メモリー容量がある場合は、バッファ・プールのバッファの数を増やします。これによって、バッファ・プールの効率が上がります。

バッファ・プールの効率が満足できるものであっても、割当てセットの有効性が低ければ、プロセスに割り当てるバッファの数を増やすことができます。これによって、割当てセットの有効性も高まります。ページが、グローバル・バッファ・プールのかわりにユーザーの割当てセットにあれば、パフォーマンスは向上します。割当てセットのサイズを大きくしすぎないように注意してください。たとえば、あるデータベースが 1000 バッファを持つグローバル・バッファ・プールを持っているとします。各プロセスに対するデフォルトの割当てセットのサイズは 30 バッファであり、20 ユーザー・プロセスがデータベースにアクセスできる必要があります。バッファ・プールに 1000 バッファとデータベースにアクセスする必要がある 20 のプロセスがある場合、デフォルトの割当てセットのサイズを 50 バッファ以上にすべきではありません。50 バッファの割当てセット・サイズが定義されている場合、データベースにアクセスする必要がある 20 ユーザー・プロセスは、それぞれ割り当てた 50 バッファを使用できます。Oracle Rdb が各プロセスに割り当てるバッファの数を決定する方法の説明は、4.1.2.2 項（4-25）を参照してください。

データベースが有効なグローバル・バッファを持っていれば、データベースのローカル・バッファが有効であれば発生するはずだったディスク I/O をページ要求の何パーセント分節約できたかを知りたいでしょう。データベースのグローバル・バッファが有効であるかわりにローカル・バッファが有効である場合、I/O 操作の結果得られたページ要求の割合を計算するには、次の式を使用してください。式の各フィールドに対して、「PIO Statistics--Data Fetches」画面と「PIO Statistics--SPAM Fetches」からのフィールドに対するカウントの総計が提供されています。

```
( 1 - ( AS: old version + in GB: old version + not found: read + : synth ) /
( AS: old version + in GB: need lock + in GB: old version +
  not found: read + : synth ) ) × 100
```

例 8-9（8-20）では、データベースのローカル・バッファが有効であったら起こるはずであったディスク I/O を節約したページ要求の割合を次に示します。

```
(1 -  171 / 1284) × 100 = 約 86.7%
```

データベースに対して、あるバッファ・プールの有効性を得るために必要なグローバル・バッファの数を決定するには、増加量を変えてバッファ・プールのサイズを大きくし、増大するグローバル・バッファ・プールに対してバッファ・プール内のページを探すページ要求の割合を計算します。

OpenVMS VAX OpenVMS Alpha

RDM$BIND_BUFFERS 論理名（A.23 項（A-11）でも説明）を設定する場合、いくつかのシステム問題を処理する必要があります。最初に、データベース・アプリケーションが実行されるアカウントに対して UAF 割当てを更新する必要があります。DIOLM は、データベース・バッファ数に等しいかまたは大きいことが必要です。ASTLM は、少なくとも DIOLM よりも 12 大きい必要があります。追加バッファに対してより多くのロックが使用される可能性があるので、ENQLM は増やす必要があります。増えたメモリー要件を取り扱うためにメモリー・パラメータ BYTLM と PGFLQUOTA は、増やす必要がある場合があります。さらに、作業セット・パラメータ WSDEFAULT、WSQUOTA および WSEXTENT は、プロセスがより多くのメモリーを持つことができるように調整が必要な可能性があります。バッファ数の増加に従って、ページ・フォルト率が大きく増加するようであれば、プロセスに対する作業セットの制限を調節すると問題を解決できます。最後に、SYSGEN パラメータ VIRTUALPAGECNT は、プロセスが追加バッファを利用する前に増加する必要があります。バッファの数が増加しても、仮想メモリーの消費が増加する結果にならなければ、VIRTUALPAGECNT 制限に達していないことになります。◆

RDM$BIND_BUFFERS がすでに最適化されている場合、RDMS$BIND_WORK_VM 論理名または RDB_BIND_WORK_VM 構成パラメータを使用すると、I/O アクティビティを削減できます（A.92 項（A-47）でも説明）。RDMS$BIND_WORK_VM と RDB_BIND_WORK_VM は、マッチング操作用にプロセスに割り当てられている仮想メモリー（VM）の総計をバイト単位で指定できるようにして、データベース・マッチング操作に対するディスク I/O 操作を削減するために使用できます。デフォルト値は、10,000 バイトです。通常、10,000 バイトの VM をすべて消費すると、Oracle Rdb は追加データ用に一時ファイルを使用します。より大きな値を指定すると、これらの一時ファイルを作成する必要性がなくなるか、低下させることができるので I/O を削減できます。

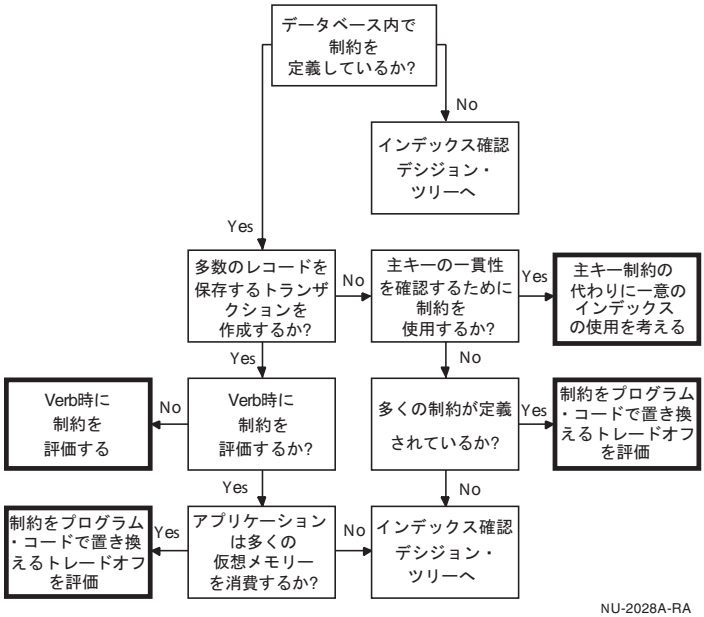
OpenVMS VAX OpenVMS Alpha

RDMS$BIND_WORK_VM の値は、UAF パラメータ PGFLQUOTA の値よりも大きくならないようにしてください。◆

8.1.3.1 制約のチェック

データベース内で制約が定義されている場合には、トランザクションを理解することで、制約がリソースをどのように使用するか理解できます（図 8-10（8-24）を参照）。たとえば、長いトランザクションが多数の行を保存する場合、仮想メモリー・リソースが消費されます。この状況では、コミット時ではなく Verb 解釈時に制約を評価できれば、消費されるメモリー・リソースは少なくなります。デフォルトでは、コミット時に制約を評価します。

図 8-10 デシジョン・ツリー : 制約のチェック

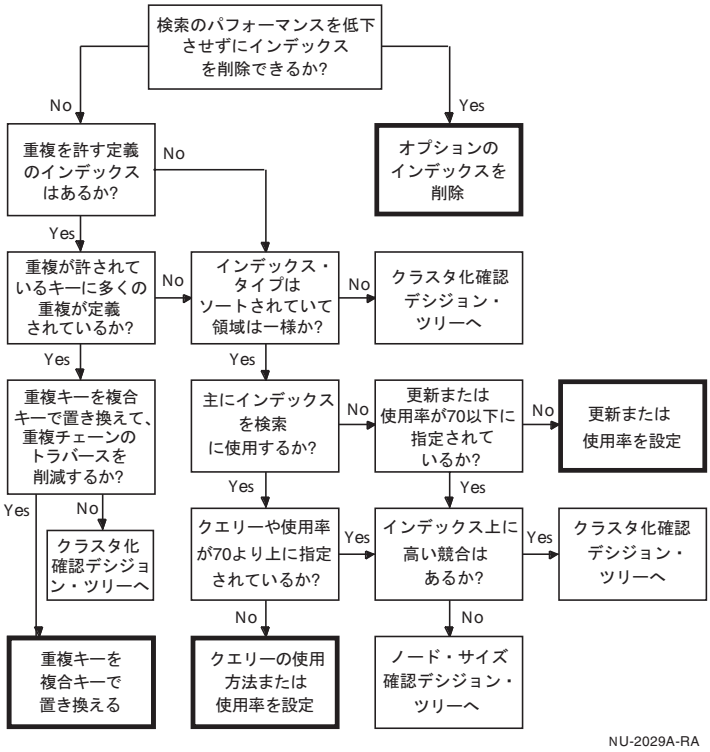


制約によってデータベース環境全体のパフォーマンスが低下していることが疑われる場合には、これを検証する最も簡単な方法は、制約を削除して、パフォーマンスがどれだけ改善するか確認することです。現行のリソースで最適に処理可能な数より多くの制約が存在する可能性があります。その他の場合として、同じ機能を実行するインデックスで制約を置き換えると、パフォーマンスが改善されます（たとえば、主キー制約を一意なインデックスで置き換える）。

8.1.3.2 インデックスのチェック

図 8-11（8-25）に、インデックスがパフォーマンスに与える影響を確認するための手順を要約します。

図 8-11 デシジョン・ツリー : インデックスのチェック



詳細は、『Oracle Rdb7 Guide to Database Design and Definition』のソート・インデックス特性の設定の項も参照してください。

テーブルに対して複数のインデックスが定義されている場合、これらのインデックスを維持するオーバーヘッドは更新のパフォーマンスを低下させます。これは、クエリー・パフォーマンスとロード・パフォーマンスのトレードオフになります。データベースにデータを迅速に格納することが重要か、それともより多数のクエリーを迅速に生成することが重要か判断する必要があります。

ロード・パフォーマンスを優先する場合で、標準レポートで使用しないインデックスが定義されている場合には、これらを削除すると、格納操作のオーバーヘッドが削減されます。イ

ンデックスに重複が多数存在する場合には、それを削除するとオーバーヘッドをさらに削減できます。標準レポートで使用しないインデックスを削除する場合、DBA は、削除したインデックスを普通に使用する一時的なクエリーをユーザーが実行しないよう配慮する必要があります。通常のインデックスを使用できない場合、これらのクエリーはテーブルを順次アクセスするため、競合の問題が発生する可能性があります。

インデックスの削除にかわる手段として、インデックスの結合があります。この方法を使用すると、インデックスを維持するための I/O の回数を削減できます。たとえば、LAST_NAME_INDEX、FIRST_NAME_INDEX および AGE_INDEX の 3 つのインデックスがあるとします。これらの 3 つのインデックスを結合して 1 つのインデックス LAST_NAME_FIRST_NAME_AGE_INDEX にして、元の 3 つのインデックスを削除すると、オーバーヘッドが減少します。1 つのインデックス定義と 1 つのインデックス構造に使用する記憶領域は、3 つのインデックスに使用する記憶領域よりも少なくてすみます。また、I/O やロックも 3 つのインデックスに対してではなく、1 つのインデックスに対して実行するだけですみます。このようなインデックスの結合は、すべてのデータベース・クエリーが、他のセグメントを指定するのではなく、結合されたインデックスの最初のインデックス・セグメントを指定する場合にのみ、実行すべきことに注意してください。

```
SQL> SELECT * FROM CONTRACTORS WHERE LAST_NAME = 'Jones';
```

結合したインデックスの最初のインデックス以外の、任意のインデックス・セグメントから検索を要求するデータベース・クエリーでは、実行パフォーマンスは低下します。

チューニング分析セッションでインデックスのオーバーヘッドを検査する場合、設計に関するもう 1 つの考慮事項に、インデックスの使用方法があります。主にダイレクト一致検索で使用する列にインデックスが定義されている場合で、クエリーに対処するソート・インデックスを定義している場合には、ハッシュ・インデックスで複合領域を作成するほうがより効率的です。たとえば、PART_NUMBER などのインデックスで、部品番号に対して厳密な一致を行って行をフェッチする場合には、このタイプの変更は意味があります。

最後に、ソート・インデックスの定義について考えます。インデックスが主にデータの効率的なロードのために使用されていることがわかっている場合には、インデックス定義に使用方法として更新を指定できます。インデックスが主にクエリーで使用される場合は、使用方法としてクエリーを指定できます。これらのオプションは、各インデックス・ノードに対して最大パーセンテージを設定します。更新は、最大パーセンテージを 70 に設定し、クエリーは最大パーセンテージを 100 に設定します。かわりに、きめ細かな制御が必要な場合には、定義時に充填パーセントを指定できます。使用方法および充填パーセントの両方が指定されている場合、使用方法のパラメータの方が優先されます。

Performance Monitor には、インデックスに関する情報を表示する画面（3.9.5.2 項（3-108）で説明）がいくつかあります。これらを使用すると、重複インデックスがどれだけ広範囲に渡って使用されているか、インデックスが挿入、削除、検索、その他同様の問題に対してどれだけ煩雑に使用されているかを判断できます。

一意なインデックスに対するカーディナリティはシステム・テーブル内で維持する必要がないので、一意なインデックスは重複インデックスより効率的であることに注意することが重要です。これで、一意なインデックスでは I/O とロッキングが減少します。

例 8-10（8-27）で示した RMU Analyze Indexes コマンドは、レベル数（最大レベル）やレコード数など、インデックス構造に関する物理情報を提供します。インデックスがより多くのレベルを持つと、Oracle Rdb は、より多くのインデックス・ノードにアクセスしてデータ・レコードを検索しなければなりません。したがって、レベルが上がれば、I/O アクティビティがさらに増えることを意味します。詳細は、3.9.5.1 項（3-97）を参照してください。

例 8-10 RMU Analyze Indexes コマンドの使用方法

```
$ RMU/ANALYZE/INDEX PRODUCTION$DB_LOGICAL
-------------------------------------------------------------------------------
---
 Indices for database  - USER18:[PRODUCTION.DATABASE]PRODUCT_DB.RDB;
-------------------------------------------------------------------------------
---
 Index PTHIST$DATE_TIME for relation PART_HISTORY duplicates allowed
Max Level: 4, Nodes: 1051, Used/Avail: 246237/418298 (59%), Keys: 20770, Records:
10349
  Duplicate nodes: 9371, Used/Avail: 149936/244528 (61%), Keys: 9371, Records: 18742
-------------------------------------------------------------------------------
---
 Index PTHIST$IDET_DATE for relation PART_HISTORY duplicates allowed
 Max Level: 2, Nodes: 18, Used/Avail: 3857/7164 (54%), Keys: 471, Records: 12
     Duplicate nodes: 910, Used/Avail: 232632/304534 (76%), Keys: 442, Records:
29079
-------------------------------------------------------------------------------
---
.
.
.
```

例 8-11（8-27）で示した RMU Analyze Placement コマンドを使用すると、インデックスが記憶領域にデータ行を配置する方法とデータ行へのアクセス方法についての情報が得られます。この情報には、データ行に到達するまでにアクセスするインデックス・レコードの最大数と平均数、データ行に到達するまでに通過したページ数の合計数、インデックスとデータ行が両方ともバッファ内に存在するどうかに関係しないバッファ・サイズの考察が含まれています。RMU Analyze Placement コマンドの詳細は、4.3.1 項（4-157）を参照してください。

例 8-11 RMUAnalyze Placement Option=Normal Command の使用方法

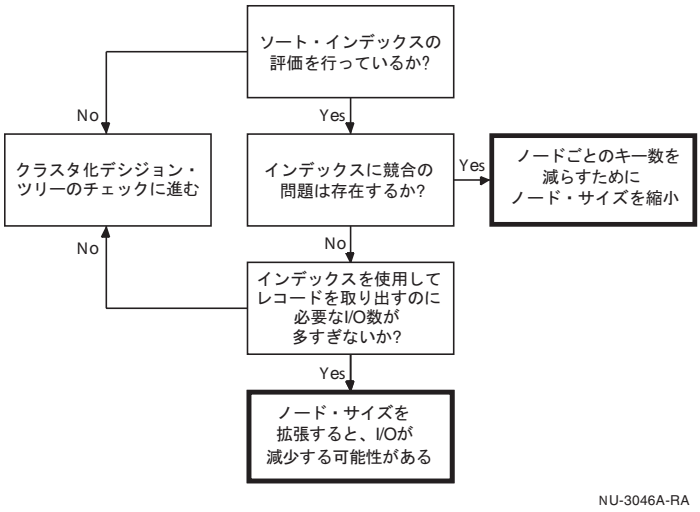
```
$ RMU/ANALYZE/PLACEMENT PRODUCTION$DB_LOGICAL PTHIST$DATE_TIME /OPTION=NORMAL
-------------------------------------------------------------------------------
 Indices for database  - USER18:[PRODUCTION.DATABASE]PRODUCT_DB.RDB;
-------------------------------------------------------------------------------
 Sorted Index PTHIST$DATE_TIME for relation PART_HISTORY duplicates not allowed
 Levels: 4, Nodes: 1051, Keys: 20770, Records: 10349
 Maximum path length --  DBkeys: 5, IO range: 1 to 5
 Average path length --  DBkeys: 4.20, IO range: 2.10 to 3.90
-------------------------------------------------------------------------------
```

この情報は、テーブルのストレージ・マップ内に指定された PLACEMENT VIA INDEX 句に対してインデックスを使用して保存された行に関して、パフォーマンスに問題があるかどうかを判断するのに便利です。見つかった問題によっては、ページ・サイズが小さすぎる、記憶領域が小さすぎる、またはデータ行やインデックス・レコードを保存する領域が不足していることを示す場合があります。

8.1.3.3 ノード・サイズのチェック

図 8-12（8-28）は、ノード・サイズの確認で使用する手順を要約しています。

図 8-12 デシジョン・ツリー : ノード・サイズのチェック



インデックス・ノードの大きさが正しいかをチェックする場合、インデックス・ノードの更新による競合の問題が存在するか、最初に判断する必要があります。競合の問題が存在する場合には、インデックスに対するノード・サイズを小さくします。ノード・サイズを小さくすると、各ノードに適合するインデックス・キー数が減少し、各更新操作でロックされるキー数も減少するので、競合の問題が緩和されます。競合の原因であるインデックスを特定することはできない場合があります。そのかわりに、競合が問題になっていないことを確認するのは可能です。そのためには、Performance Monitor の「Locking（one stat field）」画面の「stall time x 100」画面を選択して、レコード・ロックに対する統計を参照してください。この値が小さければ、インデックス内の競合は問題ではなく、インデックス・ノードは大きすぎないことを示しています。

データベース・レコードのフェッチに必要な I/O が多すぎることが問題であれば、いくつのインデックス・ノード・レベルが存在するかを判断する必要があります。また、インデック

スが独自の記憶領域に存在するか、またその領域でトランザクションあたり何回の I/O が発生しているかを判断する必要があります。インデックス・ノードのサイズを大きくすると、I/O の回数を削減できることがあります。インデックス・ノードのサイズが大きくなると、インデックス・ノード・レベルの数が減ることがあります。これによって、インデックスを使用してデータベース・レコードをフェッチする場合に必要な I/O 回数を減らすことができます。RMU Analyze Placement コマンド（例 8-11（8-27）を参照）を使用して、インデックス内のインデックス・ノード・レベルの数を表示します。次に、ソート・インデックスを使用して 1 つのレコードをフェッチする場合に必要な I/O の回数を示します。

```
Number of I/Os = Number of Node Levels + Number of Duplicate Nodes on
                 Other Pages - Number of Index Node Levels in the
   Buffer Pool + an I/O to Retrieve the Requested Record
```

5 ノード・レベルのソート・インデックスで、インデックスの使用時にそのうち 2 つがバッファ・プールに適合しており、重複ノードがない場合、式で計算すると、インデックスを使用してレコードをフェッチする場合に必要な I/O の数は 4 になります。

```
    4    =    5   +   0   -   2  +  1
```

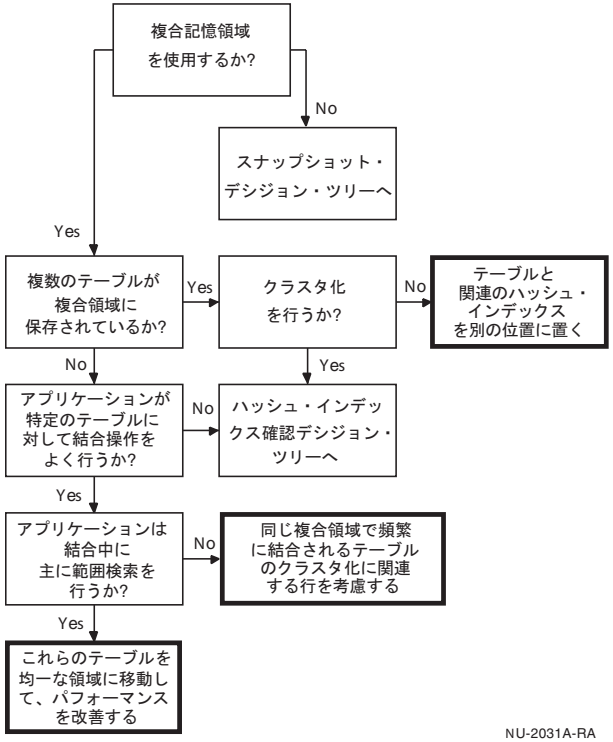
インデックスがそれ自身の記憶領域にあれば、「IO Statistics (by file)」画面を使用して、記憶領域内のトランザクションあたりの平均 I/O 回数を参照できます。値が大きい場合には、ノードのサイズを大きくすることによって、ノード・レベルの数とレコードをフェッチするのに必要な I/O を削減できます。

テーブルに対する更新がめったになく、インデックスは主にクエリーで使用されている場合には、SQL の ALTER INDEX 文または CREATE INDEX 文の PERCENT FILL 句を使用して大きな値を指定します。高い割合の充填値を指定すると、結果的にインデックスのレベルが減り、フェッチされる中間ノード数と I/O の回数が減少します。

8.1.3.4 クラスタ化のチェック

複合記憶領域を使用している場合、特定の条件のもとで、レコードのクラスタ化を使用して I/O を削減することができます（図 8-13（8-30）参照）。たとえば、アプリケーションが、複数のテーブルに渡って完全一致クエリーを使用した結合操作を頻繁に実行する場合、レコードを同じ記憶領域にクラスタ化することは、I/O 節約のための選択肢となります。言い換えると、関連するデータがいっしょにクラスタ化されているので実行する I/O 回数が少なくてすむことから、テーブル間クラスタ化は、完全一致結合操作によって検索されるデータにとってよい方法といえます。

図 8-13 デシジョン・ツリー : クラスタ化のチェック



テーブルのストレージ・マップ内のソート・インデックスに対して PLACEMENT VIA INDEX 句が指定されている場合、そのソート・インデックスを使用してレコードを特定の順序で保存できます。これで、データの順序で検索が必要なクエリーが存在する場合、パフォーマンスを改善できます。たとえば、アプリケーションが常に情報を昇順のロット番号でレポートする場合、このレコードを昇順でソートします。次に、LOT_ID 列にソート・インデックスを使用して昇順キーで保存します。これで、Oracle Rdb は、レコードをアルファベット昇順で保存します。この技術による I/O の節約は、レコードを「クラスタ化」して互いに近い位置に配置することによります。たとえば、1 つのロットが 20 パーツで構成されているとします。1 つのデータベース・ページには 10 パーツが適合し、次の形式のクエリーがしばしば実行されるとします。

```
SQL> SELECT LOT_ID, PART_ID FROM LOTS
  1>   WHERE LOT_ID > 891001;
LOT_ID            PART_ID
891001-ABCDE      00034443
```

```
891001-ABCDE      00034444
   .
   .
   .
891001-CWEEW      00355344
```

1000 ロットがデータベース内にあり、そのうちの 10 ロットがクエリーを満足し、レコードは今まで説明したようにクラスタ化されている場合、Oracle Rdb はおよそ 20 回の I/O 操作でレコードにアクセスできます（ロットあたり 2 回の I/O x 10 ロット）。このデータがクラスタ化されていなければ、最悪の場合、同じ要求に対して 2000 回の I/O 操作が発生します（レコードがばらばらでロットの各 part_id が別々のページにあり、対象ページからは非常に遠くバッファに取り込めないとします）。各ページに保存される最大レコード数が正確に予測できる場合のみ、クラスタ化を使用すべきことに注意してください。

またハッシュ・インデックスでのクラスタ化を、テーブルのストレージ・マップ内に指定された PLACEMENT VIA INDEX 句に対してインデックスを使用して行うことができます。以前に説明したように、2 つ以上のテーブルに対するダイレクト一致クエリーで結合操作が頻繁に発生するようであれば、関連するレコードを同じ記憶領域にクラスタ化することで、1 回の I/O でレコードを検索できます。

同様に、親子関係（1 対多）に基づいてレコードを検索する場合、ハッシュ・インデックスによるクラスタ化はパフォーマンスを改善します。製造では、これはロット ID やバッチ ID、およびそれに含まれるパーツの間に共通する関係です。アプリケーションにこの関係が存在して、同じ記憶領域内でレコードをいっしょにクラスタ化する場合、ファイルの拡大、ページのオーバーフローおよび断片化などによるパフォーマンスの低下を防ぐために、割当て、ページ・サイズおよび SPAM しきい値などのある記憶領域パラメータの値を注意深く計算してください。

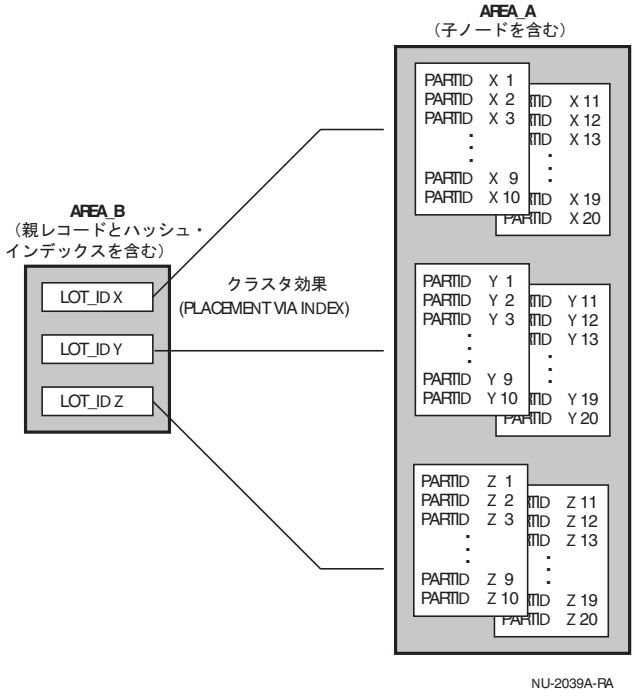
1 つの記憶領域にレコードをグループ化することが非実用的または物理的に不可能な場合、シャドウ・ページを使用すると、2 つの記憶領域に渡るクラスタ化効果を得ることができます。この方法では、親レコード（ロット・レコード）と両方のハッシュ・インデックスが 1 つの記憶領域に保存され、子ノード（パーツ・レコード）は 2 番目の領域にいっしょに保存されます。たとえば、次のように表示されます。

```
create storage map LOTID_MAP
   for WIP_LOTS
   store in AREA_B
   placement via index LOTID_HASH;
create storage map PARTID_MAP
   for PART_INFO
   store in AREA_A
   placement via index PARTID_HASH;
```

```
create unique index LOTID_HASH
   on WIP_LOTS ( LOT_ID )
   store in AREA_B
   type is HASHED;
create index PARTID_HASH
   on PART_INFO ( LOT_ID )
   store in AREA_B
   type is HASHED;
```

この方法は、記憶領域 A と記憶領域 B に、同じ相対オフセットで PART_INFO シャドウ・レコードを保存することで動作しますが、同じページ番号である必要はありません（図 8-14（8-32）を参照）。その他の例は、3.9.7.3 項（3-124）を参照してください。

図 8-14 クラスタ化におけるシャドウ・ページの使用方法

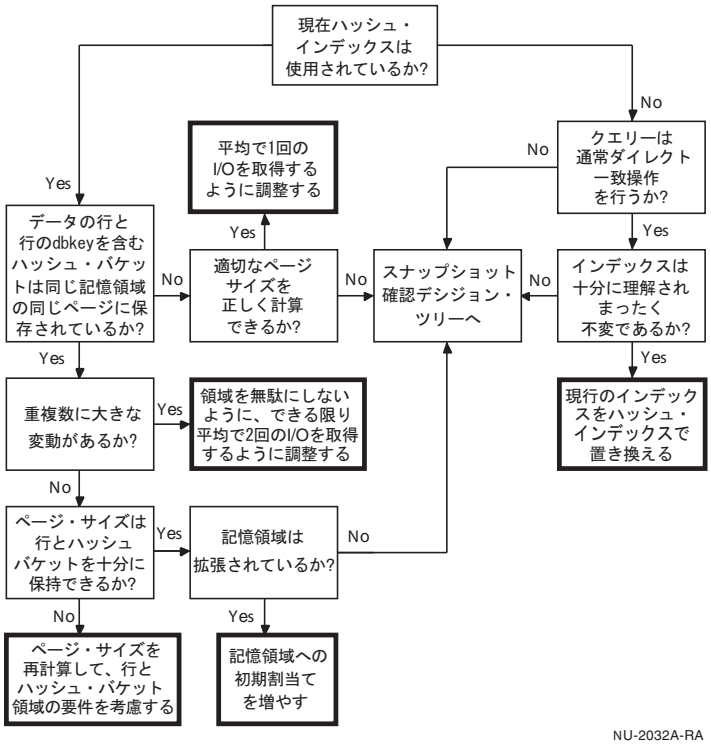


オラクル社では、その他のテーブルの保存やハッシュ・レコードまたはハッシュ・バケットを持つ複合記憶領域内のソート・インデックスはお薦めしていません。クラスタ化が目的でない場合、ハッシュ・インデックスとレコードを孤立させることで、他のデータベース・エンティティがページ上の領域を使用してしまい、注意深く計算したページ・サイズが破壊されるリスクを防止します。

8.1.3.5 ハッシュ・インデックスのチェック

図 8-15（8-33）に、パフォーマンスに対するハッシュ・インデックスの効果を確認するための手順を要約します。

図 8-15 デシジョン・ツリー : ハッシュ・インデックスのチェック



ハッシュ・インデックスまたはソート・インデックスが特別なテーブルに対して最高であるか判断するには、そのデータに対して実行するクエリーを理解する必要があります。要求が一般的にダイレクト一致クエリーであって、インデックスが非常によく特長付けられている場合は、テーブル上にハッシュ・インデックスを定義すると、データのアクセスに関して最適な結果が得られます。

ハッシュ・インデックスが定義された場合、インデックスが十分に機能するには、いくつかのパラメータが重要です。これらのパラメータには、記憶領域の初期割当て、ページ・サイズ、レコード・サイズ、ストレージ・ストラテジ、SPAM 情報、キー・サイズ、一意なキーの個数の見積りと重複数、およびハッシュ・インデックスに関連するレコードの総数の見積りがあります。

- 記憶領域の初期割当て
  ハッシュ・インデックスを使用する場合、複合記憶領域内のエクステントは深刻にパフォーマンスを損ないます。ハッシュ・キーは、新しい領域を考慮に入れて再計算されないので、挿入が行われる対象ページはおそらくいっぱいになります。レコードを挿入できる空きページを検索するために、追加 I/O が行われます。
  記憶領域が小さく、ハッシュ・インデックスを使用して大量のデータをロードする場合、記憶領域がいっぱいになるのに従って、Oracle Rdb は、対象ページから大きく離れたデータ・ページで空き領域を検索するため、より長い時間を使用するようになり、ロード時間が非常に長くなります。いくつかの小さな記憶領域に連続してロードする場合、問題はより明白になります。

- ページ・サイズ
  ハッシュ・インデックスを使用する場合、ページ・サイズを小さく見積らないことが重要です。ユーザーが保存したデータ・レコードの、圧縮前のサイズに基づいて計算を行います。
  表 8-1 （8-34）に、データ・ページの大きさの情報を示します。このトピックの詳細は、『Oracle Rdb7 Guide to Database Design and Definition』を参照してください。

表 8-1 データ・ページ上の各インデックス・レコード型に対するエントリあたりのバイト数とオーバーヘッドのバイト数の見積り

カテゴリ	エントリあたりのバイト数	総計
システム・レコード		
ハッシュ・インデックスの数 *1	a	a
システム・レコード・サイズの総計		
オーバーヘッド	4	4
最小値	6	(6 × a)+4
最大値 *2	10	(10 × a)+4
ハッシュ・バケット		
ハッシュ・バケット・エントリ・サイズの合計		
キーのサイズ *3	1	1
キーの長さ *4	k+1	k+1
オーバーヘッド / エントリ *5	12	12
総計 / エントリ *6	12+1+k+1=b	b
エントリの数 *7	c	c
オーバーヘッド / バケット *8	13	13
バケット・サイズの総計	(b × c)+13	(b × c)+13

表 8-1 データ・ページ上の各インデックス・レコード型に対するエントリあたりのバイト数とオーバーヘッドのバイト数の見積り（続き）

カテゴリ	エントリあたりのバイト数	総計
重複ノード・レコード		
重複の数 *9	d	d
エントリ / ノードの数 *10	10	
重複ノードの数 *11	(d+5)/10=e	e
オーバーヘッド / ノード *12	92	92
総計	e × 92	e × 92
ハッシュ・インデックスの総計		
	[((6 × a)+4)] OR [((10 × a)+4)]+[(b × c)+13]+[e × 92]	

*1 同じ記憶領域に格納されるハッシュ・インデックスの数。
*2 システム・レコードに対して最大サイズを想定します。
*3 キー長を保存するのに符号なしバイトを 1 バイト使用します。
*4 2 バイト長の列を含む VARCHAR（または VARYING STRING）データ型は 2 バイト長の列を含んでいますが、これはインデックス列長では無視されます。キー値は、インデックス内で空白で満たされています。NULL 値を示すのに、1 バイト（K+1 の 1）を使用します。
*5 重複カウント列（4）と dbkey ポインタ（8）。
*6 ハッシュ・バケット内の各ハッシュ・エントリのサイズは、キー・サイズの総計とキー長、およびエントリごとのオーバーヘッドの合計です。
*7 エントリは同じページをマップする任意のキー値ですが、既存のエントリの重複ではありません。
*8 レコード・タイプ（4）、オーバーフロー・バケット dbkey ポインタ（8）、フラグ列（1）の和。
*9 同じ値に対する重複レコードの数。
*10 重複ノードに入る重複エントリの最大数。
*11 最も近い整数に切り上げられた重複ノードの数。重複レコードがまったく存在しなければ、重複ノードは作成されません。
*12 1 つの重複ノードに対するオーバーヘッドの総計。

- レコード・サイズ
  レコード・サイズを決定する場合、圧縮前のユーザー保存データのサイズを使用します。レコード・サイズは、十分なページ・サイズを決定するのに必要な計算値の 1 つです。

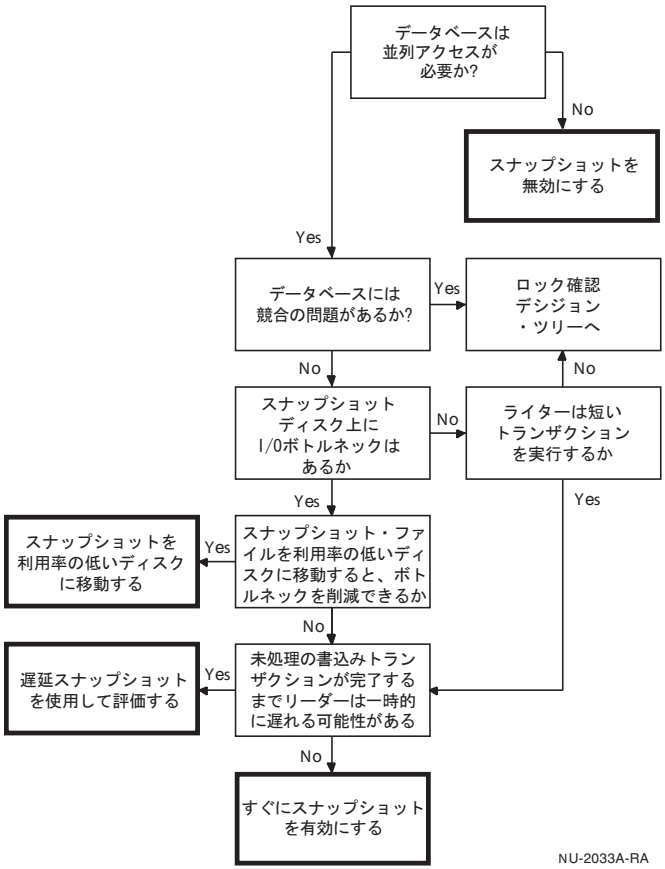
- ストレージ・ストラテジ
  次に、テーブルのストレージ・マップ内に PLACEMENT VIA INDEX 句を指定したハッシュ・インデックスを使用して、レコードを格納するための 2 つの基本的なストラテジを示します。
  - 同じ複合領域にレコードとハッシュ・バケットを保存します。
    このアプローチを使用すると、検索が平均で 1 回の I/O 操作で行えます。極端に偏った重複のない安定した情報に対して最適です。
  - 別の複合領域にレコードとハッシュ・バケットを保存します。
    これは、2 つのアプローチのうち、より保守的です。このアプローチを使用すると、ハッシュ・バケットはある記憶領域に配置され、レコードは別の記憶領域に配置されます。検索は、平均して 2 回の I/O 操作で行われます。
- SPAM 情報 --- しきい値と間隔
  - しきい値
    正しい計算が行われなければ、データベースはデータを保存する空きページを見つけるために必要以上に長く検索しなければならないか、または保存する空き領域がないことを確認するためだけにページをフェッチすることになります。2 つめのケースは、Performance Monitor の「Record Statistics」画面に表示されます。
  - 間隔
    SPAM 間隔が正しく設定されていないと、追加 I/O またはページ・ロックの増加を招くことがあります。複数の更新ユーザーが記憶領域のある部分にアクセスする場合、間隔が長いと I/O が減少し、ページ・ロッキングが増す傾向があります。
    SPAM 間隔が短ければ、ページ・ロッキング問題（Performance Monitor の「Summary Locking Statistics」画面）が減少しますが、レコードに対して空き領域を確保する I/O 操作の数は増えます。
    デフォルトのページ間隔は、216 ページです。記憶領域が大きくてページ・ロッキングが問題でない場合、この数を増やすと I/O 操作を節約できます。
    正しい SPAM しきい値と間隔は、ソート・インデックスでパフォーマンスを向上させるためにも重要であることに注意してください。
- キーのサイズ
- 一意なキーの数と重複の数の見積り
- ハッシュ・インデックスに関連したレコードの総数の見積り

最後の 3 つの項目は、ページ・サイズの総計を計算するために必要です。

8.1.3.6 スナップショットのチェック

I/O を削減する場合、スナップショットは考慮すべきもう 1 つの対象領域です（図 8-16（8-37）を参照）。

図 8-16 デシジョン・ツリー : スナップショットのチェック



情報に同時にアクセスする必要がないシングルユーザー・アプリケーションでデータベースを使用する場合、スナップショットを無効にします。必要がなければ、スナップショット・ファイルへの書込みオーバーヘッドを負う必要はありません。

デフォルトでは、スナップショット・ファイルは直接有効になっています。これは、対話型、マルチユーザー環境では、スナップショット・ファイルにとって正しい設定です。スナップショットが直接有効である場合、ライターは毎回データベースに書き込むたびにス

ナップショット・ファイルを更新するオーバーヘッドを負います。多くのユーザーがデータにアクセスする場合、直接スナップショットを使用することでパフォーマンスは通常最適になっています。このオプションは、I/O を追加して競合を削減します。

スナップショットが必要で、記憶領域があるディスクの I/O リソースがすでに限界である場合、スナップショット・ファイルをあまり利用されていないデバイスに移動することを考えてください。可能であれば、.rda、.snp、.rdb および .aij ファイルを配置すると、各ファイルタイプが別々のディスク上に存在することになります。

遅延スナップショットは、リーダーが一時的に遅れてもかまわない環境で使用できます。データベースが短いトランザクションで更新でき、リーダーがそのデータベースを常に使用しているのでなければ、遅延スナップショットから利点を得ることができます。詳細は、4.1.13 項（4-113）を参照してください。

遅延スナップショットでは、スナップショット・ファイルへの読取り / 書込みトランザクションは、書込みトランザクションが始まったときに読取り専用トランザクションが進行中の場合のみ書き込めます。読取り専用トランザクションが始まったときに書込みトランザクションがアクティブである場合、読取り専用のトランザクションに対する遅延が発生する可能性があります。読取り専用トランザクションは、アクティブな書込みトランザクションのすべてが完了するまで待つ必要があります。新しいライターが、読取り専用トランザクションが始まった後でトランザクションを開始すると、.snp ファイルに更新されようとしている列の Before-image を書き込みます。

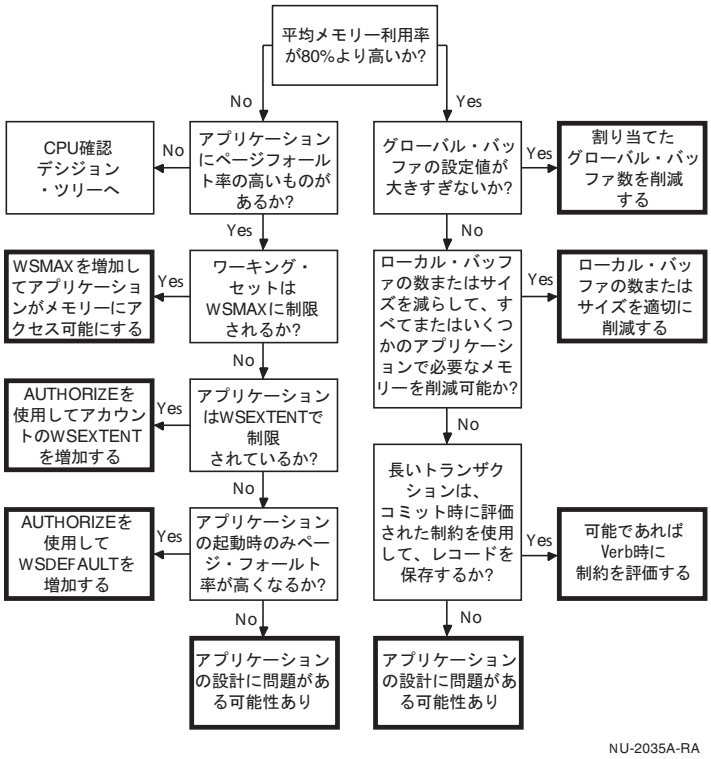
遅延を有効にしているスナップショットを使用する利点は、I/O を削減できることです。アクティブな読取り専用トランザクションがない場合、ライターはスナップショット・ファイルへの書込みを行う必要がありません。

8.2 メモリー・リソースの分析

OpenVMS VAX OpenVMS Alpha

図 8-17（8-39）は、メモリー・リソースに関連したパフォーマンス問題の識別と解決方法を支援します。

図 8-17 デシジョン・ツリー : メモリーのチェック



メモリーの使用率が 80 パーセントを超えると、アプリケーションによっては、必要なバッファが大きすぎたり、バッファを使いすぎている可能性があります。これは、アプリケーションが利用可能なメモリー・リソースで競合する原因になります。また、システムのオーバーヘッドが増えることによる速度低下の原因になる可能性があります。

アプリケーションの中にページ・フォルト率が高いものがあれば、作業セットがどこまで成長できるか作業セットの範囲制限を確認してください。SHOW PROCESS/ACCOUNTING コマンドを使用して、作業セット・サイズのピークがどれほど大きくなったかを確認します。

Authorize ユーティリティを実行すると、現在の作業セットの制限を確認できます。ピークの作業セットのサイズが WSEXTENT 値と同じであれば、そのパラメータの現在の設定によって制限される可能性があります。メモリーが利用できれば、WSEXTENT を増やすと、プロセスがメモリーにアクセスできるようになります。メモリーが必要でそのメモリーが利用できる場合のみ、作業セットの範囲 WSQUOTA と WSEXTENT の間がアプリケーションによって使用されます。アプリケーションは、システムに対して利用できるかどうかを問い

合わせなくても、WSQUOTA までの作業セットを使用できます。WSDEFAULT は、プロセスが作成されたとき、またはユーザーがアカウントにログインしたときに、プロセスが持っている作業セットの総量です。プロセスの開始時、またはユーザーが初めてログインしたときのみ、大きなページ・フォルト率を確認した場合は、WSDEFAULT の値を増やすことを考えてください。

ページ・フォルト率が高くて、アプリケーションが使用可能なメモリーにアクセスできないと感じた場合、確認すべき他のパラメータは SYSGEN 内の WSMAX です。WSMAX は、システム上の任意のプロセスが利用できる最大作業セット・サイズです。WSMAX は、共通 UAF ファイルと各ノードで異なる量のメモリー・リソースを持つクラスタ化環境における作業セットの成長を制限するためによく使用されます。しかし、WSEXTENT の値が WSMAX より大きくても、プロセスのピーク作業セットは WSMAX で指定した値によって制限されます。この状況では、WSMAX の値を増やして WSEXTENT よりも大きい値にすると決定することも可能です。

```
$ RUN SYSGEN
SYSGEN>  USE ACTIVE
SYSGEN>  SHOW WSMAX
Parameter Name             Current   Default   Minimum   Maximum Unit  Dynamic
--------------             -------   -------   -------   ------- ----  -------
WSMAX                         8200      1024        60    100000 Pages
SYSGEN>  EXIT
$
```

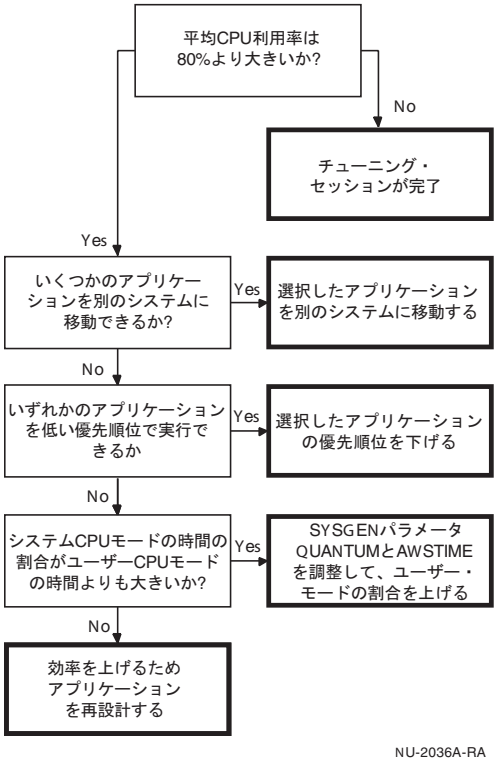
◆

8.3 CPU リソースの分析

OpenVMS VAX OpenVMS Alpha

CPU リソースの制限を処理することは困難です。図 8-18（8-41）に従うと、リソースを評価するステップを識別できます。オプションには、いくつかのアプリケーションを別のシステムに移動したり、マシンにアクセスできるユーザーの数を制限するものが含まれています。バッチと対話型処理が同じ優先順位で発生したら、バッチ・アクティビティの優先順位を下げて、対話型ユーザーに CPU リソースの最初のチャンスを与えることができます。このオプションを使用して、書込みトランザクションの優先順位を下げる前に十分考慮してください。優先順位の高いアプリケーションが必要なリソースを優先順位の低いアプリケーションが保持している場合、競合問題の増加をもたらします。

図 8-18 デシジョン・ツリー : CPU のチェック



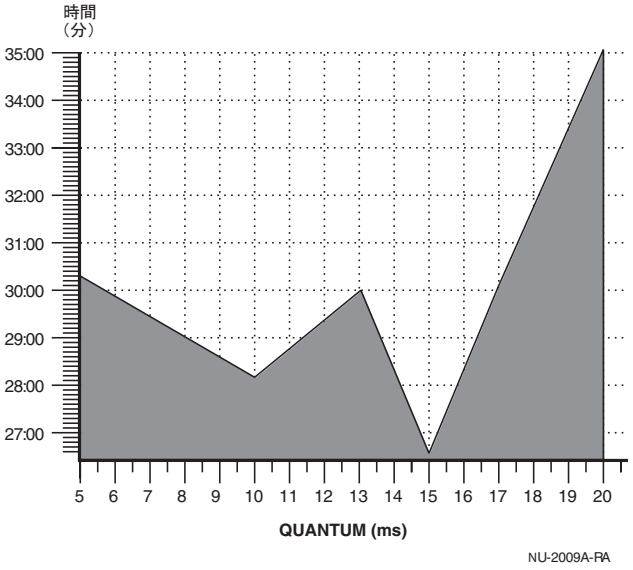
CPU リソースの制限を削減するために使用できるその他のオプションに、SYSGEN パラメータの QUANTUM と AWSTIME の変更があります。QUANTUM は、次のプロセスが順番を得る前にシステムによって与えられたプロセスが使用できる CPU 時間です。AWSTIME は、作業セットの調整を試みる前にシステムが待つ時間の長さです。QUANTUM を 10 ミリ秒（ms）より低く設定すると、その他の特別なパラメータである IOTA は、プロセスが終点に早く到達しすぎないようにデフォルトの 2ms よりも低い 1ms に下げられます。IOTA は、I/O 終了時にプロセスの残っている QUANTUM から差し引く ms の数値で、I/O 集中型プロセスに対して CPU を解放させます。これらのパラメータは動的なので、システムを停止したり、起動したりすることなく変更でき、その効果もすぐに測定できます。これらのパラメータを効率的に使用するため、優先順位の高いアプリケーションを考慮して、I/O 集中型か CPU 集中型（バッファではなく、ディスク I/O）かを判断する必要があります。I/O 集中型アプリケーションは、QUANTUM の値が小さいほど利点が得られる傾向があります。CPU 集中型アプリケーションは、高めの QUANTUM 設定から利点が得られます。これらのパラ

メータは、アプリケーション単位ベースで変更できません。変更はシステム上のすべてのプロセスに影響を与えるので、これらのパラメータを変更するときは注意してください。

これらのパラメータを最適に設定して得られたリソースの節約は、オペレーティング・システムのオーバーヘッドを削減して得たものです。このような節約は、ユーザー関連の作業で使用されます。QUANTUM がデフォルトよりも低く設定されている場合、AWSTIME の値は QUANTUM の 2 倍に設定することをお薦めします。QUANTUM がデフォルトよりも増加している場合、AWSTIME は QUANTUM と同じに設定してください。どちらの場合も、作業セットの調整の確認によるシステムのオーバーヘッドは、削減されます。CPU 集中型アプリケーションの場合、これらのアプリケーションは CPU 時間の次の割当てまで完了できませんが、QUANTUM が大きければすぐに完了できる可能性があります。これは、応答時間を改善し、プログラム・ステータスを次の順番まで一時的に保存するために必要なオーバーヘッドを削減します。

QUANTUM チューニング研究（図 8-19（8-42）および図 8-20（8-43））からの図式は、この情報を説明するために含まれています。図 8-19（8-42）は、対象となる作業負荷応答時間における QUANTUM の効果を示しています。

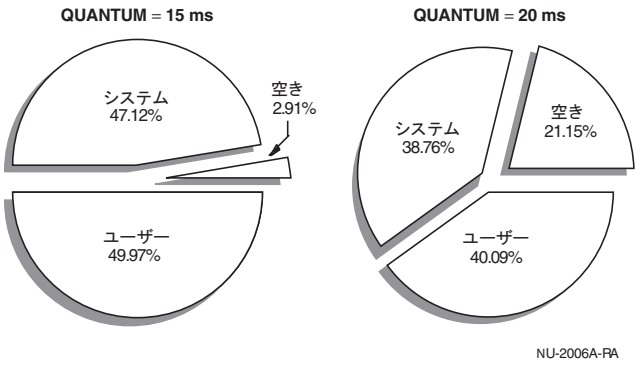
図 8-19 作業負荷応答時間におけるの QUANTUM の効果



これらの応答時間の改善は、CPU 時間の非常に小さな変動として現れます。また、メモリーと作業セットは、毎回のテストで同じです。CPU 利用率のわずかな変化は、各間隔に対する新しい作業セットの調整周期の部分に原因がある可能性があります。これによって、

アプリケーション作業セットの応答が異なり、ダイレクト I/O またはページ・フォルト率のマイナーチェンジの原因になります。図 8-20（8-43）は、QUANTUM 設定が 15ms の場合と 20ms の場合の CPU モードを比較します。

図 8-20 QUANTUM 設定が 15ms と 20ms の場合の CPU モードの比較



◆

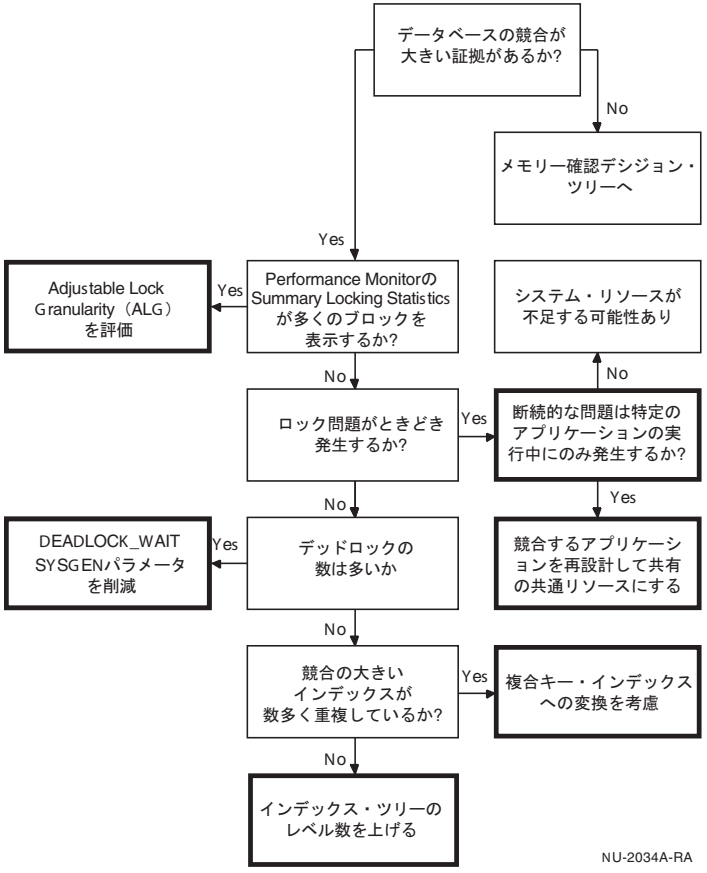
8.4 ロック・リソースの分析

ロックは、データベース・アプリケーションの応答時間に大きな影響を与える可能性があります。ロックは、アプリケーションが応答時間の遅延を示す断続的なパフォーマンス問題の原因になりますが、CPU、I/O およびメモリー・リソースはまだ利用できます。

ロック問題には、多くの原因があります。アプリケーションは、ロックに対して拘束性が高い共有リソース、または多くのユーザーが競合するリソースを持っている可能性があります。これらの問題が、遂行可能な最大データベーススループットを制限します。

図 8-21（8-44）は、ロック問題を確認するために使用できるステップを要約します。

図 8-21 デシジョン・ツリー : ロックのチェック



データベース内での競合が明白であれば、たとえば、多くのロック・リソースが使い果たされるか、多くのアプリケーションが Performance Monitor の「Stall Messages」画面で停止しているか、「Performance Monitor Summary Locking Statistics」画面（(例 8-12（8-45））が多くのブロッキング AST を示している場合、Adjustable Lock Granularity(ALG) を停止すると、Oracle Rdb が最低レベルのロックをすぐに使用できるのでブロッキング AST の数が減少します。（ブロッキング AST の増加は、キャリーオーバー・ロック最適化にも関連しています。詳細は、3.8.3.2 項（3-69）を参照してください。）ALG の情報は、3.8.5 項（3-76）を参照してください。アプリケーション・コード内の DECLARE TRANSACTION と SET TRANSACTION 文を検査して、最大の並列性が許可されていることを確認してください。トランザクションがテーブルを読み取るだけであれば、トランザクションを読取り専用に、

アクセス・モードを共有読取りに設定します（トランザクションに対して SQL SET TRANSACTION READ ONLY RESERVING テーブル名 FOR SHARED READ 文を指定します）。SQL では、デフォルトは読取り / 書込みおよび共有書込みです。したがって、アプリケーションがデフォルトを使用している場合、トランザクションが読取り専用トランザクションであればリソースに対して必要以上の保護的なロックを持ちます。この問題の兆候には、予測以上に増加した競合と低下した並列性が含まれています。

例 8-12 Performance Monitor の「Summary Locking Statistics」画面

```
Node: MYNODE          Oracle Rdb V7.0-00 Performance Monitor 30-MAY-1996 14:59:39
Rate: 3.00 Seconds          Summary Locking Statistics       Elapsed: 00:30:14.24
Page: 1 of 1     USER18:[PRODUCTION.DATABASE]PRODUCT_DB.RDB;1       Mode: Online
-------------------------------------------------------------------------------
statistic.........      rate.per.second.............. total....... average......
name..............      max..... cur..... avg....... count....... per.trans....
locks requested               78       52       36.8        59013         115.3
 rqsts not queued              1        0        0.2          349           0.7
 rqsts stalled                 2        0        0.2          390           0.8
 rqst timeouts                 0        0        0.0            0           0.0
 rqst deadlocks                0        0        0.0            0           0.0
locks promoted                60       47       14.9        23840          46.6
 proms not queued              0        0        0.0           29           0.1
 proms stalled                 0        0        0.0           20           0.0
 prom timeouts                 0        0        0.0            0           0.0
 prom deadlocks                0        0        0.0            0           0.0
locks demoted                101       78       22.7        36417          71.1
locks released               190       35       36.4        58324         113.9
blocking ASTs                  4        0        0.4          615           1.2
stall time x100                5        0        1.0         1630           3.2
invalid lock block             0        0        0.0            0           0.0
-------------------------------------------------------------------------------
Exit Graph Help Menu Options Pause Reset Set_rate Time_plot Unreset Write X_plot
```

ロックの問題が断続的であれば、アプリケーションはおそらくインデックスを使用するのではなく、データベースに順次アクセスしています。アプリケーションが共有書込みモードで読取り / 書込みトランザクションを開始して、順次アクセスする場合、共有書込みロックは保護付き書込み（PW）に変えられます。これは、テーブル全体をロックする効果があり、断続的なロック問題の原因になる可能性があります。アプリケーションが予期したインデックスを使用しているかどうかを見るためには、RDMS$DEBUG_FLAGS 論理名または RDB_DEBUG_FLAGS 構成パラメータを使用します。次の例に示すように、名前に S を設定すると、検索ストラテジに結果が表示されます。

```
$ SQL
SQL> ATTACH 'FILENAME mf_personnel';
SQL> SELECT JOB_TITLE FROM JOBS WHERE JOB_CODE = "VPSD";
.
.
```

```
.
Conjunct        Get     Retrieval sequentially of relation JOBS
 JOB_TITLE
 Vice President
1 row selected
SQL>EXIT
$
```

RDMS$DEBUG_FLAGS 論理名と RDB_DEBUG_FLAGS 構成パラメータの使用方法の詳細は、付録 C（C-1）を参照してください。

その他にも、パフォーマンスの低下が発生している領域には、極端なデッドロックがある可能性があります。デッドロックは、アプリケーションまたは貧困なアプリケーション設計によって発生することがあります。

OpenVMS VAX OpenVMS Alpha

アプリケーションがデッドロックしがちであれば、SYSGEN パラメータの DEADLOCK_WAIT を調整すると状況をより迅速に検出できます。デフォルト値は、10 秒です。環境がデッドロックしがちであれば、パラメータを低くします。

パラメータを低くすることで、デッドロックはより迅速に検出され、アプリケーションはほとんど遅延しなくなります。このコストは、ロック・タイムアウト・キューを検索するオーバーヘッドを増やします。ほとんどデッドロックが発生しない場合、このパラメータを増やすと、真のデッドロック状況が長い間検出されない状況が発生しますが、非常にまれにしか存在しないデッドロックを検索するオーバーヘッドを削減します。割り当てられるロックが多くなると、キューの検索に必要な時間が増え、デッドロック条件を検索するオーバーヘッドが増します。

```
$ RUN SYSGEN
SYSGEN>  USE ACTIVE
SYSGEN>  SHOW DEADLOCK_WAIT
Parameter Name              Current   Default   Minimum   Maximum Unit  Dynamic
--------------              -------   -------   -------   ------- ----  -------
DEADLOCK_WAIT                    10        10         0        -1 Seconds     D
SYSGEN> EXIT
```

DEADLOCK_WAIT パラメータを低くすると、デッドロックをより迅速に検出する支援のみを行うことに注意してください。◆

データベース内で多くのデッドロックが発生する場合、データベースやアプリケーションを変更することで、いくつかを修正できます。デッドロックを修正するためにデータベースに対して行えるいくつかの変更には、ソート・インデックス・ノードのサイズの削減またはソート・インデックスからハッシュ・インデックスへの変換があります。アプリケーションに対して行える変更には、カーソルのみを使用した更新またはトランザクションに対する制限モード（書込み保護モードなど）があります（制限モードを使用するとデッドロックを修正する可能性がありますが、トランザクションの実行中に競合問題が発生する可能性があります）。場合によっては、デッドロックは避けることができません。

A

Oracle Rdb の論理名と構成パラメータ

この付録では、Oracle Rdb 論理名と構成パラメータを説明し、データベースのパフォーマンスを改善するため、いつどのように使用するかを説明します。

「Hot Standby」オプションに対して指定する論理名と構成パラメータは、この付録では説明しません。「Hot Standby」オプションに特定の論理名と構成パラメータの説明は、『Oracle Rdb7 and Oracle CODASYL DBMS: Guide to Hot Standby Databases』を参照してください。

A.1 RDB$CHARACTER_SET

OpenVMS VAX OpenVMS Alpha

Oracle Rdb で使用する代替キャラクタ・セットを定義できます。有効な代替キャラクタ・セットには、次のものが含まれています。

- DEC_KANJI、日本
- DEC_HANZI、中国
- DEC_HANGUL、韓国
- DEC_HANYU、台湾

例 A-1（A-2）は、キャラクタ・セットを日本語に設定しています。

例 A-1 RDB$CHARACTER_SET 論理名の使用方法

```
$ DEFINE RDB$CHARACTER_SET DEC_KANJI
```

RDB$CHARACTER_SET 論理名は、将来のリリースに対して推奨されません。◆

A.2 RDB$LIBRARY と RDB_LIBRARY

外部関数などの外部ルーチン・イメージの格納に使用可能な保護付きライブラリを指定します。オラクル社は、論理名 RDB$LIBRARY または構成パラメータ RDB_LIBRARY によって参照される保護付きライブラリを使用して、パブリックまたは繊細な外部ルーチン・イメージを管理するよう推奨します。

OpenVMS VAX OpenVMS Alpha

RDB$LIBRARY をシステム論理名テーブル内で実行モード論理名として定義する必要があります。外部ルーチン・イメージが保護付き領域にある場合、LOCATION 句と CREATE FUNCTION 文内の WITH SYSTEM LOGICAL_NAME TRANSLATION 句で明示的なファイル名を持つ RDB$LIBRARY 論理名を指定することで必要としているイメージが使用されているかどうかを確認できます。◆

A.3 RDB$RDBSHR_EVENT_FLAGS

OpenVMS VAX OpenVMS Alpha

OpenVMS では、起動時に次のイベント・フラグを RDB$SHARE に割り当てます。

- Stall イベント・フラグ
- Common イベント・フラグ
- リモート読込みイベント・フラグ
- リモート書込みイベント・フラグ

これらのイベント・フラグは、内部 DECdtm 処理で使用します。共通イベント・フラグのデフォルト値は、31 です。その他のイベント・フラグ値は、デフォルトで LIB$GET_EF システム・サービスによって割り当てられます。

RDB$RDBSHR_EVENT_FLAGS 論理名を使用すると、デフォルトのイベント・フラグ値を上書きできます。これは、イベント・フラグを他の目的で使用して、Oracle Rdb イベント・フラグを特定の範囲や値のセットに制限したい場合に便利です。

0 から 63 の範囲にある 4 つの整数値を含む文字列として論理名の値を指定します。引用符で文字列を囲み、カンマで整数を区切ります。これらの値は、前のリストで示したイベント・フラグと同じ順番で、イベント・フラグに割り当てられます。

RDB$RDBSHR_EVENT_FLAGS 論理名を変換する間にエラーが発生したり、指定したイベント・フラグ値が範囲外であった場合、Oracle Rdb は LIB$GET_EF システム・サービスによって割り当てられたデフォルトのイベント・フラグ値を使用します。例 A-2（A-3）では、RDB$RDBSHR_EVENT_FLAG 論理名の使用方法を示しています。

例 A-2 RDB$RDBSHR_EVENT_FLAGS 論理名の使用方法

```
$ DEFINE RDB$RDBSHR_EVENT_FLAGS "63,62,61,60"
```

RDB$RDBSHR_EVENT_FLAGS 論理名は、LMN$DCL_LOGICAL 論理表を使用して変換します。◆

A.4 RDB$REMOTE_BUFFER_SIZE および SQL_NETWORK_BUFFER_SIZE

デフォルトでは、ネットワーク転送のバッファ・サイズは 4,096 バイトです。RDB$REMOTE_BUFFER_SIZE 論理名または SQL_NETWORK_BUFFER_SIZE 構成パラメータを使用して、ネットワーク転送のデフォルトのバッファ・サイズを変更できます。最小値は 500 バイト、最大値はシステムのリソースと割当てのみによって制限されています。500 バイト未満の値を指定すると、Oracle Rdb はデフォルト値である 4,096 バイトを使用します。

大きなデータ・ブロックをデータベースに転送する場合、アプリケーションを実行する前にネットワーク・バッファのサイズを増やしておくと便利です。バッファ・サイズを増やすと、大量に転送する場合に行うネットワーク I/O 操作の回数が削減できます。

Compaq Tru64 UNIX

Compaq Tru64 UNIX 上では、バッファ・サイズを 10,000 バイトに設定するには、例 A-3（A-3）で示した行を構成ファイル内に含めてください。

例 A-3 SQL_NETWORK_BUFFER_SIZE 構成パラメータの使用方法

```
SQL_NETWORK_BUFFER_SIZE 10000
```

◆

クライアント・ノードとサーバー・ノードのどちらかまたは両方の構成ファイルに対して、ネットワーク・バッファ・サイズを設定できます。これらの値が特別なクライアントおよびサーバーの組合せに対して異なる場合、2 つのうちの小さいほうの値が使用されます。

OpenVMS VAX OpenVMS Alpha

RDB$REMOTE_BUFFER_SIZE 論理名は、LNM$DCL_LOGICAL 論理テーブルを使用して変換されます。◆

A.5 RDB$REMOTE_MULTIPLEX_OFF および SQL_NETWORK_NUMBER_ATTACHES

RDB$REMOTE_MULTIPLEX_OFF 論理名と SQL_NETWORK_NUMBER_ATTACHES 構成パラメータは、同じノードにアクセスする複数のリモート・データベースで使用するリモート・サーバー・プロセスの数を制御します。

RDB$REMOTE_MULTIPLEX_OFF 論理名に任意の値を定義するか、SQL_NETWORK_NUMBER_ATTACHES パラメータに値 1 を設定する場合、リモート・データベース・アクセスのそれぞれで、リモート・ノード上にある独自の RDB_SERVER プロセスが必要です。SQL_NETWORK_NUMBER_ATTACHES 構成パラメータに大きな値を割り当てると、そのノードに対するリモート・データベースのアクセスはその数値まで、リモート・ノード上の 1 つの RDB_SERVER プロセスを共有できます。SQL_NETWORK_NUMBER_ATTACHES に対するデフォルト値は、10 です。

Compaq Tru64 UNIX

Compaq Tru64 UNIX 上では、ネットワーク・アタッチを 20 に増やすには、例 A-4（A-4）で示した行を構成ファイル内に含めてください。

例 A-4 SQL_NETWORK_NUMBER_ATTACHES 構成パラメータの使用方法

```
SQL_NETWORK_NUMBER_ATTACHES 20
```

◆

OpenVMS VAX OpenVMS Alpha

RDB$REMOTE_MULTIPLEX_OFF 論理名は、LNM$DCL_LOGICAL 論理テーブルを使用して変換されます。◆

A.6 RDB$ROUTINES および RDB_ROUTINES

外部ルーチン・イメージの場所を指定します。CREATE FUNCTION 文でロケーション句を指定しないか、または DEFAULT LOCATION 句を指定する場合、SQL は RDB$ROUTINES 論理名または RDB_ROUTINES 構成パラメータをデフォルト・イメージ位置として使用します。

A.7 RDBVMS$CREATE_DB および RDB_CREATE_DB

論理名 RDBVMS$CREATE_DB または構成パラメータ RDB_CREATE_DB を定義すると、データベースの作成を制限できます。

OpenVMS VAX OpenVMS Alpha OpenVMS では、rights 識別子を同じ名前で定義する必要があります。◆

注意：RDBVMS$CREATE_DB または RDB_CREATE_DB を定義すると、その他にインストールされている Oracle 製品やサード・パーティ製品は、Oracle Rdb を使用して Oracle Rdb データベースを作成できません。したがって、それらの製品のユーザーが Oracle Rdb データベースを作成する必要があるときは、論理名の割当てを削除するか、構成ファイルからパラメータを削除する必要があります。

OpenVMS VAX OpenVMS Alpha OpenVMS で、Oracle Rdb データベースの作成を制限するには、例 A-5（A-5）で示したように、まず RDBVMS$CREATE_DB 論理名を定義します。

例 A-5 RDBVMS$CREATE_DB 論理名の使用方法

```
$ DEFINE/SYSTEM/EXECUTIVE RDBVMS$CREATE_DB RESTRICT_DB
```

次に、rights 識別子 RDBVMS$CREATE_DB を使用して、どのユーザーが SQL CREATE DATABASE 文を使用してデータベースを作成できるかを制御します。詳細は、『Oracle Rdb7 Guide to Database Design and Definition』を参照してください。◆

A.8 RDM$BIND_ABS_LOG_FILE および RDB_BIND_ABS_LOG_FILE

RDM$BIND_ABS_LOG_FILE 論理名を使用するか、RDB_BIND_ABS_LOG_FILE 構成パラメータを使用して、After-image ジャーナル・バックアップ・サーバー（ABS）ログ・ファイル用のファイル名を定義します。

OpenVMS VAX OpenVMS Alpha LNM$SYSTEM_TABLE テーブルでこの論理名を定義する必要があります。◆

A.9 RDM$BIND_ABS_OVERWRITE_ALLOWED および RDB_BIND_ABS_OVERWRITE_ALLOWED

RDM$BIND_ABS_OVERWRITE_ALLOWED 論理名または RDB_BIND_ABS_OVERWRITE_ALLOWED 構成パラメータを使用して、After-image ジャーナル・バックアップ・サーバーが、上書きされた AIJ をリセットできるかどうかを示します。デフォルト値 0（ゼロ）は、ABS が上書きされたジャーナルをリセットできないことを示し、値 1 は、リセットできることを示します。

A.10 RDM$BIND_ABS_OVERWRITE_IMMEDIATE および RDB_BIND_ABS_OVERWRITE_IMMEDIATE

論理名 RDM$BIND_ABS_OVERWRITE_IMMEDIATE または構成パラメータ RDB_BIND_ABS_OVERWRITE_IMMEDIATE を使用すると、RDM$BIND_ABS_OVERWRITE_ALLOWED または DB_BIND_ABS_OVERWRITE_ALLOWED が有効であれば、ジャーナルをすぐにリセットするべきであるかどうかを示すことができます。

デフォルト値 0（ゼロ）は、AIJ をすぐにリセットする必要はないこと、値 1 は、AIJ をすぐにリセットする必要があることを示しています。

A.11 RDM$BIND_ABS_QUIET_POINT および RDB_BIND_ABS_QUIET_POINT

論理名 RDM$BIND_ABS_QUIET_POINT または構成パラメータ RDB_BIND_ABS_QUIET_POINT を使用して、After-image ジャーナル・バックアップ・サーバー（ABS）が静止ポイント After-image ジャーナル・バックアップを実行できるかどうかを示します。

デフォルト値 0（ゼロ）は、静止ポイント・バックアップが実行されないことを、値 1 は実行されることを示します。

データベースにアクセスするすべてのノードで論理名と構成パラメータを定義する必要があります。

OpenVMS VAX OpenVMS Alpha

次の例で示すように RDM$BIND_ABS_QUIET_POINT をシステム論理名表内の実行モード論理名として定義する必要があります。

```
$ DEFINE/SYSTEM/EXEC RDM$BIND_ABS_QUIET_POINT 1
```

◆

RDM$BIND_ABS_QUIET_POINT または RDB_BIND_ABS_QUIET_POINT に割り当てた値は、現行ノード上のすべてのデータベースに影響を与えることに注意してください。

A.12 RDM$BIND_ABW_ENABLED および RDB_BIND_ABW_ENABLED

RDM$BIND_ABW_ENABLED 論理名または RDB_BIND_ABW_ENABLED 構成パラメータを使用して非同期バッチ書込み操作を有効または無効にできます。デフォルト値 1 は、非同期バッチ書込み操作が有効であることを、値 0（ゼロ）は無効であることを示しています。

Compaq Tru64 UNIX

Compaq Tru64 UNIX 上では、非同期バッチ書込み操作を無効にするには、例 A-6（A-7）で示した行を構成ファイル内に含めてください。

例 A-6 RDB_BIND_ABW_ENABLED 構成パラメータの使用方法

```
RDB_BIND_ABW_ENABLED 0
```

◆

非同期バッチ書込み操作は、デフォルトで有効です。

非同期バッチ書込み操作の情報は、3.2.5 項（3-25）を参照してください。

A.13 RDM$BIND_AIJ_CHECK_CONTROL_RECS および RDB_BIND_AIJ_CHECK_CONTROL_RECS

RDM$BIND_AIJ_CHECK_CONTROL_RECS 論理名と RDB_BIND_AIJ_CHECK_CONTROL_RECS 構成パラメータは、AIJ キャッシュ形成の間にレコードの制御を確認するかどうかを示します。デフォルト値 1 は、Oracle Rdb が制御レコードをチェックすることを、値 0（ゼロ）は Oracle Rdb は制御レコードをチェックしないことを示します。

A.14 RDM$BIND_AIJ_EMERGENCY_DIR および RDB_BIND_AIJ_EMERGENCY_DIR

RDM$BIND_AIJ_EMERGENCY_DIR 論理名または RDB_BIND_AIJ_EMERGENCY_DIR 構成パラメータを使用して、緊急 AIJ の位置を指定します。この論理名または構成パラメータは、緊急 AIJ を作成するデバイスとディレクトリのみを指定する必要があります。定義されていれば、RDM$BIND_AIJ_EMERGENCY_DIR 論理名または RDB_BIND_AIJ_EMERGENCY_DIR 構成パラメータは、現行ノード上にあるすべてのデータベースに適用されることに注意してください。

OpenVMS VAX OpenVMS Alpha

OpenVMS では、論理名には、非システム隠し論理定義を含むことはできません。また、LNM$SYSTEM_TABLE 論理名テーブル内に存在しなければなりません。◆

A.15 RDM$BIND_AIJ_IO_MAX および RDB_BIND_AIJ_IO_MAX

RDM$BIND_AIJ_IO_MAX 論理名と RDB_BIND_AIJ_IO_MAX 構成パラメータを使用すると、最大 AIJ グループ・コミットの I/O バッファ・サイズを上書きできます。デフォルト・バッファ・サイズは 127 ブロックです。

A.16 RDM$BIND_AIJ_IO_MIN および RDB_BIND_AIJ_IO_MIN

RDM$BIND_AIJ_IO_MIN 論理名と RDB_BIND_AIJ_IO_MIN 構成パラメータを使用すると、最小 AIJ グループ・コミットの I/O バッファ・サイズを上書きできます。デフォルト・バッファ・サイズは 8 ブロックです。

A.17 RDM$BIND_AIJ_STALL および RDB_BIND_AIJ_STALL

RDM$BIND_AIJ_STALL 論理名および RDB_BIND_AIJ_STALL 構成パラメータは、After-image ジャーナル（.aij）ログ・ファイルへコミット・レコードをサブミットした後、トランザクションが待機する時間をミリ秒で定義します。この待機時間に、グループ・コミット操作内に大量のトランザクションを入れることができます。

種々のベンチマークに基づいて、最適な待機時間である 50 ミリ秒がデフォルトの値として設定されています。しかし、真に最適な値は、個々のアプリケーションによって異なります。アプリケーションが更新集中型でなければ、値が小さいほどパフォーマンスをわずかに改良する可能性があります。アプリケーションが更新集中型であれば、値が大きいほどパフォーマンスがわずかによいという結果が得られます。

注意: オラクル社は、絶対的に必要な場合を除いて、デフォルト値を変更しないようにお薦めします。

デフォルト値は 50 ミリ秒です。最小値は 0 ミリ秒で、最大値は 1000 ミリ秒（1 秒）です。

A.18 RDM$BIND_AIJ_SWITCH_GLOBAL_CKPT および RDB_BIND_AIJ_SWITCH_GLOBAL_CKPT

RDM$BIND_AIJ_SWITCH_GLOBAL_CKPT 論理名または RDB_BIND_AIJ_SWITCH_GLOBAL_CKPT 構成パラメータを使用して、AIJ 切り替えが発生した後で、グローバル・チェックポイントを実行するかどうかを指定できます。デフォルト値 1 は、グローバル・チェックポイントが実行されることを示し、値 0（ゼロ）は実行されないことを示します。

A.19 RDM$BIND_ALS_CREATE_AIJ および RDB_BIND_ALS_CREATE_AIJ

RDM$BIND_ALS_CREATE_AIJ 論理名または RDB_BIND_ALS_CREATE_AIJ 構成パラメータを使用すると、AIJ 切替え操作が一時停止状態になった場合に、ALS サーバーが緊急 AIJ を作成するかどうかを指定できます。

AIJ 切替え操作が完了できなければ、利用可能な AIJ がないので、データベースは "AIJ suspended" 状態になります。この状態の間に、DBA は新しい AIJ を追加するか、データベースのバックアップを取ることができますが、その他のすべての AIJ 関連アクティビティは、AIJ が利用できるようになるまで、一時的に停止します。

AIJ 停止期間中は、DBR 起動はデータベースをシャットダウンします。DBR は、常に AIJ にコミット・レコードまたはロールバック・レコードを書き込むので必要です。DBR が読取り専用トランザクションに対して実行されたとしても、データベースはシャットダウンされることに注意してください。

デフォルト値 0（ゼロ）は、ALS が AIJ を作成しないことを示しており、値 1 は、ALS が AIJ を作成しようとすることを示しています。OpenVMS では、この論理名は LNM$SYSTEM_TABLE 論理名テーブルにある必要があります。

RDM$BIND_ALS_CREATE_AIJ 論理名の値が 1 に設定された場合、ALS は、以前の AIJ をテンプレートとして使用し、緊急 AIJ を作成しようとします。これは、緊急 AIJ が切り替えられた AIJ と同じディレクトリに、同じアロケーションで作成されたことを意味します。ディスク領域が不足しているか、またはその他のエラーが発生した場合、データベースは "AIJ suspended" 状態になり、DBA はその状況を解決する必要があります。

警告 : 緊急 AIJ は、一時 AIJ ではありません。AIJ を DCL から削除しないでください。必ず SQL または RMU の構文のみを使用して緊急 AIJ を削除してください。手動で緊急 AIJ を削除すると、データベースがシャットダウンされます。

ALS は、オペレータ通知機能を使用して DBA に緊急 AIJ が作成されたことを通知します。さらに、RMU Dump Header コマンドからの出力を使用して、ALS サーバー・ジャーナルによって作成された AIJ を識別できます。Performance Monitor も識別された緊急 AIJ をハイライトします。

緊急 AIJ には、次に示す特徴があります。

- 緊急 AIJ は、すべての点で通常の AIJ です。これは、AIJ 切替えが一時停止状態にならないように、ALS プロセスによって作成されます。AIJ 作成中のアプリケーション・プロセスの障害による DBR 実行では、データベースのシャットダウンが発生することはありません。

- 緊急 AIJ の名前は EMERGENCY_XXX です。ここで、XXX は一意な名前を作成するために使用する 16 文字の並びです。
- 緊急 AIJ の作成はジャーナル化されていません。これはホット・スタンバイ・データベースの複製がアクティブである間は、緊急 AIJ の作成は可能ではないことを意味しています。
- 緊急 AIJ の緊急状態を削除する方法はありません。

A.20 RDM$BIND_APF_DEPTH および RDB_BIND_APF_DEPTH

Oracle Rdb がプロセスの非同期プリフェッチを実行する際のバッファの数（深さ）を指定します。

OpenVMS VAX OpenVMS Alpha

例 A-7（A-10）は、Oracle Rdb が 1 つのプロセスに対して 5 つのバッファを非同期にプリフェッチを指定する方法を示しています。

例 A-7 RDM$BIND_APF_DEPTH 論理名の使用方法

```
$ DEFINE RDM$BIND_APF_DEPTH 5
```

◆

Oracle Rdb 非同期プリフェッチ機能の使用方法の詳細は、3.2.4 項（3-23）を参照してください。

A.21 RDM$BIND_APF_ENABLED および RDB_BIND_APF_ENABLED

論理名 RDM$BIND_APF_ENABLED および構成パラメータ RDB_BIND_APF_ENABLED を使用すると、非同期プリフェッチ操作を無効または有効にできます。デフォルト値 1 は、非同期プリフェッチ操作が有効であることを、値 0（ゼロ）は無効であることを示しています。

Compaq Tru64 UNIX

非同期プリフェッチ操作を無効にするには、例 A-8（A-10）で示した行を構成ファイル内に含めます。

例 A-8 RDB_BIND_APF_ENABLED 構成パラメータの使用方法

```
RDB_BIND_APF_ENABLED 0
```

◆

非同期プリフェッチ操作はデフォルトで有効です。

非同期プリフェッチ操作の詳細は、3.2.4 項（3-23）を参照してください。

A.22 RDM$BIND_BATCH_MAX および RDB_BIND_BATCH_MAX

RDM$BIND_BATCH_MAX 論理名および RDB_BIND_BATCH_MAX 構成パラメータは、バッチ書込みまたは非同期バッチ書込み操作の一部として、データベースに書き込まれるライブ・データ・キャッシュ・バッファの数を定義します。バッチ書込み操作は、新しいキャッシュ・バッファが必要であっても、利用できるバッファがない場合に発生します。この状況を調整するため、Oracle Rdb は 1 つ以上の既存のキャッシュ・バッファをデータベースに書き込んで、交換の候補にします。非同期バッチ書込み操作は、プロセスの置換用のバッファの LRU キューの末尾に保持するクリーン・バッファの数が、プロセスに対して RDM$BIND_CLEAN_BUF_CNT 論理名または RDB_BIND_CLEAN_BUF_CNT 構成パラメータで指定したクリーン・バッファの数よりも少ない場合に発生します。

RDM$BIND_BATCH_MAX 論理名または RDB_BIND_BATCH_MAX 構成パラメータが定義されていない場合、大きな I/O バーストの結果発生するバッチ書込み操作は、システムのパフォーマンスに予期せぬ影響を与えます。この論理名または構成パラメータを、ユーザーに割り当てられるキャッシュ・バッファの数よりも小さい値に設定することにより、I/O バッファのサイズを制御できます。これにより、システムの I/O 処理の予測と均一化できます。

この論理名と構成パラメータに最適な値は、ユーザーごとに割り当てられるキャッシュ・バッファの数、データベース内の記憶領域の数、ディスク・ドライブのタイプと速度、CPU ロード、アプリケーションの作業負荷など、いくつかの要因に依存します。値の設定が小さすぎると、バッチ書込み操作が急増します。値の設定が大きすぎると、大きな I/O バースト・パターンが発生します。

この論理名と構成パラメータの値は、スナップショット・ページで使用するバッチ書込み操作のサイズまたはコミットやチェックポイント操作で使用されるバッチのサイズに影響を与えません。

RDM$BIND_BATCH_MAX 論理名と RDB_BIND_BATCH_MAX 構成パラメータのデフォルト値は、ユーザーに割り当てられるキャッシュ・バッファの数です。指定した数が、使用されたバッファの数より大きくてもこれが最大値です。最小値は、1 です。

Oracle Rdb がユーザーの割当てセットのサイズを決定する方法の詳細は、4.1.2.2 項（4-25）を参照してください。

A.23 RDM$BIND_BUFFERS および RDB_BIND_BUFFERS

RDM$BIND_BUFFERS 論理名または RDB_BIND_BUFFERS 構成パラメータを使用すると、Oracle Rdb が実行時に各データベース・ユーザーに対して割り当てるバッファの数を変更できます。デフォルトでは、Oracle Rdb は NUMBER OF BUFFERS IS パラメータで指定された値を割り当てます。RDM$BIND_BUFFERS および RDB_BIND_BUFFERS の動作は、ローカル・バッファを使用しているかグローバル・バッファを使用しているかで違います。

- ローカル・バッファが有効である場合（デフォルト）、RDM$BIND_BUFFERS および RDB_BIND_BUFFERS に対して、2 から 524288 の間の値を指定できます。
- グローバル・バッファが有効である場合、RDM$BIND_BUFFERS または RDB_BIND_BUFFERS で指定したバッファの数は、USER LIMIT IS パラメータによって設定された値を超えることはできません。USER LIMIT IS 値は、プロセスがグローバル・バッファ・プールから割り当てられる最大数を定義し、この値を超えた RDM$BIND_BUFFERS 値または RDB_BIND_BUFFERS 値を上書きします。

4.1.2 項（4-19）は、RDM$BIND_BUFFERS 論理名と RDB_BIND_BUFFERS 構成パラメータをローカルおよびグローバル・バッファで、いつどのように使用するかを説明しています。

RDM$BIND_BUFFERS 論理名と RDB_BIND_BUFFERS 構成パラメータは、特定のアプリケーションのチューニングに対して強力なツールです。たとえば、初期ロード・プログラムに対してデフォルトを超えるバッファの数を定義して、データベースがロードできるようにし、順次データベース検索を行えるようにします。RDM$BIND_BUFFERS または RDB_BIND_BUFFERS を使用すると、オフピーク時間に実行するバッチプログラムに対して、より多くのバッファを割り当てることができますが、通常の作業時間中に使用するユーザーに対するデフォルトのバッファ数は保持します。

OpenVMS VAX OpenVMS Alpha

例 A-9（A-12）は、論理名を定義するプロセスに対して 100 バッファを割り当てます。

例 A-9 RDM$BIND_BUFFERS 論理名の使用方法

```
$ DEFINE RDM$BIND_BUFFERS 100
```

必要な権限があれば、より大きなプロセスのグループに影響を与えるグループまたはシステム論理名として RDM$BIND_BUFFERS を定義できます。◆

A.24 RDM$BIND_BUFOBJ_ENABLED

OpenVMS Alpha

Oracle Rdb には、OpenVMS Alpha システムからのデータベース・アクセスに対するパフォーマンス拡張機能があります。この機能は、OpenVMS のバッファ・オブジェクト機能を使用して、Oracle Rdb ローカル・バッファを物理メモリーにロックします。メモリーに Oracle Rdb ローカル・バッファをロックすると、SMP パラレル化とスケーリングが増加し、OpenVMS の作業負荷がなくなることで I/O のパフォーマンスが改善されます。

この機能を使用するには、次の項目を実行する必要があります。

- Alpha ユニプロセッサまたは SMP システム上の OpenVMS Alpha Version 6.1 以降のバージョンで Oracle Rdb V6.1 以降のバージョンを実行する。
- 論理名 RDM$BIND_BUFOBJ_ENABLED に任意の値を定義する。
- ローカル・バッファ・プール内のバイト数によって OpenVMS ユーザー割当て BYTLM を増やす。たとえば、ローカル・バッファの数が 1000 で、バッファ・サイズが 8 ブ

ロックであれば、次の式を使用して BYTLM パラメータの値を決定します。
1000 × (8 × 512) = 4,096,000 バイト

メモリーが制約されているシステム上で、この機能を使用しないでください。バッファ・オブジェクトとして定義された OpenVMS ページは、ページアウトされたり、スワップアウトされたりしますが、物理メモリー内に存在する必要があります。この物理メモリーは、イメージが終了するまで他のプロセスで利用できなくなります。

OpenVMS バッファ・オブジェクトの Oracle Rdb の利用率は、グローバル・バッファを有効にしたデータベース上では利用できません。◆

A.25 RDM$BIND_CBL_ENABLED および RDB_BIND_CBL_ENABLED

RDM$BIND_CBL_ENABLED 論理名と RDB_BIND_CBL_ENABLED 構成パラメータは、粗いバッファ・ロックを有効にするかどうかを示します。デフォルト値 0（ゼロ）は粗いバッファ・ロックが無効であることを、値 1 は、有効であることを示します。

A.26 RDM$BIND_CKPT_BLOCKS および RDB_BIND_CKPT_BLOCKS

RDM$BIND_CKPT_BLOCKS 論理名または RDB_BIND_CKPT_BLOCKS 構成パラメータを使用すると、チェックポイントが発生した後の AIJ ブロックの数を指定できます。デフォルト値は 0（ゼロ）ブロックです。

A.27 RDM$BIND_CKPT_TIME および RDB_BIND_CKPT_TIME

RDM$BIND_CKPT_TIME 論理名または RDB_BIND_CKPT_TIME 構成パラメータを使用すると、チェックポイントが発生した後の時間の総計を秒単位で指定できます。デフォルト値は 0 です。

A.28 RDM$BIND_CKPT_TRANS_INTERVAL および RDB_BIND_CKPT_TRANS_INTERVAL

RDM$BIND_CKPT_TRANS_INTERVAL 論理名または
RDB_BIND_CKPT_TRANS_INTERVAL 構成パラメータを使用して、プロセスごとのチェックポイント間隔の値を定義できます。デフォルトでは、高速コミット処理が有効であれば、AIJ ブロック・サイズの制限に到達、または間隔の制限を超過、のどちらかが最初に発生し

たときにプロセスのチェックポイントが実行されます。
RDM$BIND_CKPT_TRANS_INTERVAL 論理名と RDB_BIND_CKPT_TRANS_INTERVAL 構成パラメータは、チェックポイント・トリガーとしてトランザクションの数を使用します。したがって、RDM$BIND_CKPT_TRANS_INTERVAL または RDB_BIND_CKPT_TRANS_INTERVAL が 10 になるように定義した場合、仮にブロックサイズまたは時間の制限が到達する前にトランザクションの制限が到達した場合は、10 トランザクションをコミットした後で論理名または構成パラメータを持つプロセスはチェックポイントを行うように定義します。詳細は、4.1.5.2 項（4-93）を参照してください。

A.29 RDM$BIND_CLEAN_BUF_CNT および RDB_BIND_CLEAN_BUF_CNT

RDM$BIND_CLEAN_BUF_CNT 論理名または RDB_BIND_CLEAN_BUF_CNT 構成パラメータを使用すると、プロセスの置換用バッファの LRU キューの末尾に保持するクリーン・バッファの数を指定できます。この論理名と構成パラメータは、プロセスで発生する I/O ストールの数を減らすことができる非同期バッチ書込み機能の一部として使用されます。RDM$BIND_CLEAN_BUF_CNT および RDB_BIND_CLEAN_BUF_CNT に対するデフォルト値は 5 です。これは、プロセスの置換用バッファの LRU キューの末尾に 5 つのクリーン・バッファを保持することを意味しています。

RDM$BIND_BATCH_MAX 論理名と RDB_BIND_BATCH_MAX 構成パラメータも、非同期バッチ機能とともに使用できます。RDM$BIND_BATCH_MAX および RDB_BIND_BATCH_MAX の詳細は、A.22 項（A-11）を参照してください。

非同期バッチ書込み操作の詳細は、3.2.5 項（3-25）を参照してください。

A.30 RDM$BIND_COMMIT_STALL および RDB_BIND_COMMIT_STALL

RDM$BIND_COMMIT_STALL 論理名と RDB_BIND_COMMIT_STALL 構成パラメータは、トランザクションがグループ・コミット・プロセスとなろうとした後に待機する時間をミリ秒で指定します。この待機時間により、グループ・コミット操作内に大量のトランザクションを入れることができます。

種々のベンチマークに基づいて、最適な待機時間である 50 ミリ秒がデフォルト値として設定されています。しかし、真に最適な値は、特定のアプリケーションによって異なります。アプリケーションが更新集中型でなければ、値が小さいほどパフォーマンスをわずかに改良する可能性があります。アプリケーションが更新集中型であれば、値が大きいほどパフォーマンスがわずかによいという結果が得られます。

注意：オラクル社は、必要な場合を除いて、デフォルト値を変更しないようにお薦めします。

デフォルト値は 50 ミリ秒です。最小値は 0 ミリ秒で、最大値は 1000 ミリ秒（1 秒）です。

A.31 RDM$BIND_DAPF_DEPTH_BUF_CNT および RDB_BIND_DAPF_DEPTH_BUF_CNT

RDM$BIND_DAPF_DEPTH_BUF_CNT 論理名と RDB_BIND_DAPF_DEPTH_BUF_CNT 構成パラメータを使用すると、物理領域からプリフェッチできるバッファの数を指定できます。デフォルト値は、データベース・ユーザー向けに定義したバッファの数の 1/2 です。

A.32 RDM$BIND_DAPF_ENABLED および RDB_BIND_DAPF_ENABLED

論理名 RDM$BIND_DAPF_ENABLED と構成パラメータ RDB_BIND_DAPF_ENABLED を使用すると、検出された非同期プリフェッチ（Detected Asynchronous PreFetch：DAPF）読取り操作を無効にできます。DAPF 機能は、物理領域が順次読み取られている場合、物理領域の読み取られる次のページを予測します。前の予測と読取りページが一致したら、Oracle Rdb は、実際の（順次）読取り要求よりも先に非同期にページをプリフェッチします。

デフォルト値 1 は検出された非同期プリフェッチ操作が有効であることを示し、値 0（ゼロ）は無効であることを示します。

Compaq Tru64 UNIX

非同期プリフェッチ操作を無効にするには、例 A-10（A-15）で示した行を構成ファイル内に含めます。

例 A-10 RDB_BIND_DAPF_ENABLED 構成パラメータの使用方法

```
RDB_BIND_DAPF_ENABLED 0
◆
```

DAPF 機能は、主に複合フォーマット領域および次の場合のパフォーマンスを改善します。

- レコードが連続するページに保存される場合に多数のレコードを挿入
- RMU Verify コマンド、または DIO（またはレコード）層を渡すことで、Oracle Rdb の PIO（またはページ）層にアクセスするその他の操作を実行する場合。
- ノードが隣接する集合にある場合、隣接するソート・インデックス・リーフ・ノードを左から右に移動する場合。

非同期プリフェッチ読取り操作はデフォルトで有効です。

A.33 RDM$BIND_DAPF_START_BUF_CNT および RDB_BIND_DAPF_START_BUF_CNT

論理名 RDM$BIND_DAPF_START_BUF_CNT と構成パラメータ RDB_BIND_DAPF_START_BUF_CNT を使用すると、検出された非同期プリフェッチ読取り操作が開始する前に物理領域から順次アクセスできるバッファの数を指定できます。

A.34 RDM$BIND_HRL_ENABLED および RDM_BIND_HRL_ENABLED

RDM$BIND_HRL_ENABLED 論理名と RDM_BIND_HRL_ENABLED 構成パラメータは、ホールド読取りロックが有効であるかどうかを示します。デフォルト値 0 はホールド読取りロックが無効であることを、値 1 は有効であることを示しています。

A.35 RDM$BIND_LOCK_TIMEOUT_INTERVAL および RDB_BIND_LOCK_TIMEOUT_INTERVAL

RDM$BIND_LOCK_TIMEOUT_INTERVAL 論理名または RDB_BIND_LOCK_TIMEOUT_INTERVAL 構成パラメータを定義することで、デフォルトの待機間隔を指定できます。

Compaq Tru64 UNIX

例 A-11（A-16）は、Compaq Tru64 UNIX における構成パラメータの定義方法を示しています。

例 A-11 RDB_BIND_LOCK_TIMEOUT_INTERVAL 構成パラメータの使用方法

```
RDB_BIND_LOCK_TIMEOUT_INTERVAL 15
```

例 A-11（A-16）では、待機間隔を 15 秒に指定しています。待機間隔は秒単位で表されていますが、間隔はおおよその値であることに注意してください。待機間隔に指定した時間の総計は、アプリケーションによって異なります。◆

OpenVMS VAX　OpenVMS Alpha

しかし、一般的なガイドラインとして、SYSGEN パラメータ DEADLOCK_WAIT で指定された値よりも大きな値を使用します。DEADLOCK_WAIT パラメータの詳細は、4.4.2 項（4-182）を参照してください。◆

アプリケーション特有の待機時間を指定するには、SQL SET TRANSACTION 文の WAIT 句を使用します。WAIT 句で指定された間隔は、論理名または構成パラメータによって指定された待機間隔を上書きします。

WAIT 句、RDM$BIND_LOCK_TIMEOUT_INTERVAL 論理名または RDB_BIND_LOCK_TIMEOUT_INTERVAL 構成パラメータで設定できる時間の総計の上限として機能するデータベース全体のデフォルトのロック・タイムアウト間隔を設定できます。このデータベース全体のロック・タイムアウト間隔は、SQL CREATE または ALTER DATABASE 文の LOCK TIMEOUT INTERVAL IS n SECONDS パラメータで設定できます。データベース全体のロック・タイムアウト間隔を指定するならば、その間隔はさらに高い値を持つその他のロック・タイムアウト間隔を上書きします。

データベース全体のロック・タイムアウトは、タイムアウト間隔を決定する上限と同じようにデフォルトとして使用されます。たとえば、データベース定義として、CREATE DATABASE 文で LOCK TIMEOUT INTERVAL IS 25 SECONDS を指定し、そのデータベースのユーザーが SET TRANSACTION WAIT 30 を指定するか、論理名 RDM$BIND_LOCK_TIMEOUT_INTERVAL を変更するか、構成パラメータ RDB_BIND_LOCK_TIMEOUT_INTERVAL を 30 に設定すると、SQL は 25 秒の間隔を使用します。

詳細は、『Oracle Rdb7 Guide to Distributed Transactions』および『Oracle Rdb7 SQL Reference Manual』を参照してください。

A.36 RDM$BIND_MAX_DBR_COUNT および RDB_BIND_MAX_DBR_COUNT

RDM$BIND_MAX_DBR_COUNT 論理名と RDB_BIND_MAX_DBR_COUNT 構成パラメータは、データベース・モニターが同時に実行するデータベース・リカバリ（DBR）プロセスの最大数を定義します。この論理名と構成パラメータは、グローバル・バッファが有効でないデータベースのみに適用されます。グローバル・バッファを使用するデータベースでは、同時に開始するリカバリ・プロセスは 1 つのみです。

OpenVMS VAX OpenVMS Alpha

OpenVMS では、この論理名は LNM$SYSTEM_TABLE 論理名テーブルで定義されている必要があります。次の例では、リカバリ・プロセスの最大数を 20 に制限しています。

```
$ DEFINE/SYSTEM RDM$BIND_MAX_DBR_COUNT 20
```

◆

A.37 RDM$BIND_OPTIMIZE_AIJ_RECLEN および RDB_BIND_OPTIMIZE_AIJ_RECLEN

RDM$BIND_OPTIMIZE_AIJ_RECLEN 論理名か RDB_BIND_OPTIMIZE_AIJ_RECLEN 構成パラメータを定義すると、RMU Optimize After_Journal コマンドのパフォーマンスを改善できます。

一般に、オラクル社では次の指針に従って After-image ジャーナル最適化の全体的なパフォーマンスを改善することをお薦めします。

- 常に RDMS$BIND_SORT_WORKFILES 論理名か RDB_BIND_SORT_WORKFILES 構成パラメータを使用して、使用する作業ファイルの数を指定します。
- 常に SORTWORKn 論理名を使用して、使用されていないできるだけ高速なデバイス上の一時作業ファイルのファイル名を指定します。
- 同じデバイス上には 2 つ以上のファイルを置かないでください。
- 作業ファイル・デバイスに大きな空き領域がある場合は、使用する作業ファイルの数を少なくします。空き領域が限られている場合は、多くの作業ファイルを使用します。
- 出力を追跡する必要がない場合は、RMU Optimize After_Journal コマンドで Trace 修飾子を使用しないでください。トレース出力を実行すると、バッファ I/O やダイレクト I/O 操作の数が増大し、経過時間全体が長くなります。
- Log 修飾子を使用すると、OPTRECLEN に関するメッセージが表示されます。
- OPTRECLEN のメッセージの出力 +10% を RDM$BIND_OPTIMIZE_AIJ_RECLEN 論理名か RDB_BIND_OPTIMIZE_AIJ_RECLEN 構成パラメータの値として使用します。最小値は 512、最大値は 4096 です。
- RDM$BIND_OPTIMIZE_AIJ_RECLEN または RDB_BIND_OPTIMIZE_AIJ_RECLEN の値よりも大きな DDL レコードや .aij レコードの多い .aij ファイルは最適化しないでください。この種のレコードには即時ソートとフラッシュ操作が必要です。この操作を実行すると、莫大な費用がかかり、大きな出力ファイルが生成されます。レコードの正確な数は、アプリケーションと入力 After-image ジャーナル全体のサイズによって大きく変わります。

A.38 RDM$BIND_RCACHE_INSERT_ENABLED および RDB_BIND_RCACHE_INSERT_ENABLED

RDM$BIND_RCACHE_INSERT_ENABLED 論理名または RDM$BIND_RCACHE_INSERT_ENABLED 構成パラメータを使用すると、行キャッシュ内に行を挿入できるかどうかを示します。デフォルト値 1 は行がキャッシュ内に挿入できることを示し、値 0 は挿入できないことを示します。

A.39 RDM$BIND_RCACHE_RCRL_COUNT および RDB_BIND_RCACHE_RCRL_COUNT

RDM$BIND_RCACHE_RCRL_COUNT 論理名または RDB_BIND_RCACHE_RCRL_COUNT 構成パラメータを使用すると、予約する行キャッシュ・スロットの数を指定できます。Oracle Rdb は、デフォルトで 20 行のキャッシュ・スロットを予約します。

A.40 RDM$BIND_RCS_BATCH_COUNT および RDB_BIND_RCS_BATCH_COUNT

RDM$BIND_RCS_BATCH_COUNT 論理名と RDB_BIND_RCS_BATCH_COUNT 構成パラメータは、行キャッシュ・サーバー（Row Cache Server：RCS）が 1 つのバッチで削除する行の数を定義します。

デフォルト値は 3000 行です。最小値は 1 行で、最大値は 1,000,000 行です。

OpenVMS VAX OpenVMS Alpha

OpenVMS では、この論理名は LNM$SYSTEM_TABLE 論理名テーブルで定義されている必要があります。◆

A.41 RDM$BIND_RCS_CHECKPOINT および RDB_BIND_RCS_CHECKPOINT

RDM$BIND_RCS_CHECKPOINT 論理名または RDB_BIND_RCS_CHECKPOINT 構成パラメータを使用すると、行キャッシュ・サーバーにチェックポイントを実行するように指示できます。デフォルト値 1 は、チェックポイントが実行されることを示し、値 0 は実行されないことを示します。RCS プロセスがチェックポイントを実行する場合、行キャッシュ内にあるマークされたすべてのコールド・レコードをそれぞれのバックアップ格納域ファイルに書き込みます。

A.42 RDM$BIND_RCS_CKPT_BUFFER_CNT および RDB_BIND_RCS_CKPT_BUFFER_CNT

RDM$BIND_RCS_CKPT_BUFFER_CNT 論理名または RDB_BIND_RCS_CKPT_BUFFER_CNT 構成パラメータを使用すると、チェックポイント操作の間、行キャッシュ・サーバープロセスによって、シングル・バッチとして検査されるバッファの数を示します。バッファ・カウントのデフォルト値は 15 バッファです。

A.43 RDM$BIND_RCS_LOG_FILE および RDB_BIND_RCS_LOG_FILE

RDM$BIND_RCS_LOG_FILE 論理名または RDB_BIND_RCS_LOG_FILE 構成パラメータを使用して、行キャッシュ・サーバーログ・ファイル用のファイル名を定義します。

OpenVMS VAX OpenVMS Alpha

LNM$SYSTEM_TABLE テーブルでこの論理名を定義する必要があります。◆

A.44 RDM$BIND_RCS_MAX_COLD および RDB_BIND_RCS_MAX_COLD

RDM$BIND_RCS_MAX_COLD 論理名または RDB_BIND_RCS_MAX_COLD 構成パラメータを使用すると、マーク・レコードの数を指定でき、これを上回ると行キャッシュ・サーバーが削除を開始します。RCS プロセスが削除を実行する場合、行キャッシュ内にあるマークされたすべてのコールド・レコードをそれぞれの記憶領域に書き込みます。

A.45 RDM$BIND_RCS_MIN_COLD および RDB_BIND_RCS_MIN_COLD

RDM$BIND_RCS_MIN_COLD 論理名または RDB_BIND_RCS_MIN_COLD 構成パラメータを使用すると、マーク解除レコードの数を指定でき、これを下回ると行キャッシュ・サーバーが削除を完了します。RCS プロセスが削除を実行する場合、行キャッシュ内にあるマーク付きのすべてのコールド・レコードをそれぞれの記憶領域に書き込みます。

A.46 RDM$BIND_RCS_SWEEP_INTERVAL および RDB_BIND_RCS_SWEEP_INTERVAL

RDM$BIND_RCS_SWEEP_INTERVAL 論理名または RDB_BIND_RCS_SWEEP_INTERVAL 構成パラメータを使用すると、削除する間隔を分で示します。デフォルトの削除間隔は 1 分です。

A.47 RDM$BIND_READY_AREA_SERIALLY および RDB_BIND_READY_AREA_SERIALLY

RDM$BIND_READY_AREA_SERIALLY 論理名または
RDB_BIND_READY_AREA_SERIALLY 構成パラメータを使用すると、Oracle Rdb は、ロックが要求された順序で論理および物理領域にロック要求を付与します。

ロック・マネージャには、ロック要求に応える WAIT キューおよび CONVERSION キューという 2 つのキューがあります。プロセスがロックを要求した場合、2 つのうちの 1 つが発生します。プロセスは、要求されたモードのロックをすぐに得るか、（他のプロセスがすでに互換性のないモードでロックしている場合）プロセスをリソースに対する NL ロックを付与するための WAIT キューに入れるかのどちらかを行います（リソースに対してプロセスが要求したモードにかかわらず）。プロセスが NL モードでロックを取得したら、NL モード・ロックを望みの（元来の要求である）モードのロックに変換するために WAIT キューから CONVERSION キューに移動されます。CONVERSION キュー上では、現在許可されているモードと互換性があるロック要求は、キュー上の他の互換性のないロックよりも優先されます（つまり、現在許可されているモードと互換性がある CONVERSION キュー上のロック要

求は、CONVERSION キュー上でより長く待っている互換性のないロック要求よりも前に許可されます)。CONVERSION キュー内のロック要求は、WAIT キュー内のロック要求よりも常に高い優先順位を得ています。

プロセス・ロック要求を WAIT キューから CONVERSION キューに急いで移動すると、ロック・スタベーションの可能性が減少します。ロック・スタベーションは、プロセスが要求したモードのロックが許可されない状況を説明する用語です。たとえば、CONVERSION キュー上の多くのユーザーが読込み領域および非読込み領域を持っているとします。WAIT キュー上の互換性がないロック要求が CONVERSION キューに迅速に移動されなければ、WAIT キュー上のプロセスはなかなかロックが得られなくなります。(CONVERSION キュー内のロック要求は、WAIT キュー内のロック要求よりも優先順位が与えられているため)。

現在許可されているモードと互換性がある CONVERSION キューで要求されているロックは、CONVERSION キュー上の互換性のないロック要求よりも優先権を与えられています。これは、CONVERSION キュー上に互換性のある要求が多くあり、互換性のない要求が少なければ、互換性のない要求はなかなかロックを得られなくなります。この状況で発生する可能性があるロック・スタベーションを避けるには、RDM$BIND_READY_AREA_SERIALLY または RDB_BIND_READY_AREA_SERIALLY に 1 を定義します。これを設定すると、Oracle Rdb は CONVERSION キューを番号順にします。したがって、RDM$BIND_READY_AREA_SERIALLY または RDB_BIND_READY_AREA_SERIALLY が 1 として定義した場合、Oracle Rdb は論理領域および物理領域に対する CONVERSION キューのロック要求をロック要求が行われた順番で許可します。

OpenVMS VAX OpenVMS Alpha

例 A-12（A-21）は、ロック・スタベーションを防ぐために RDM$BIND_READY_AREA_SERIALLY 論理名を定義する方法を示しています。

例 A-12 RDM$BIND_READY_AREA_SERIALLY 論理名の使用方法

```
$ DEFINE/SYSTEM/EXECUTIVE RDM$BIND_READY_AREA_SERIALLY 1
```

◆

A.48 RDM$BIND_RUJ_ALLOC_BLKCNT および RDB_BIND_RUJ_ALLOC_BLKCNT

RDM$BIND_RUJ_ALLOC_BLKCNT 論理名または RDB_BIND_RUJ_ALLOC_BLKCNT 構成パラメータを使用して、.ruj ファイルの初期サイズをブロック単位で定義できます。デフォルトの .ruj ファイルのサイズは、127 ブロックです。

A.49 RDM$BIND_RUJ_EXTEND_BLKCNT および RDB_BIND_RUJ_EXTEND_BLKCNT

RDM$BIND_RUJ_EXTEND_BLKCNT 論理名または RDB_BIND_RUJ_EXTEND_BLKCNT 構成パラメータを使用して、データベースを使用する各プロセスに対する .ruj ファイルを事前に拡張できます。たとえば、新しいブロックのカウントを 1 から 65535 の間の値で定義することも、デフォルトの 127 ブロックを受け入れることもできます。

Compaq Tru64 UNIX

Compaq Tru64 UNIX 上では、.ruj ファイルを事前に拡張するために例 A-13（A-22）で示した行を構成ファイル内に含めてください。

例 A-13 RDB_BIND_RUJ_EXTEND_BLKCNT 構成パラメータの使用方法

```
RDB_BIND_RUJ_EXTEND_BLKCNT 1000
```

◆

A.50 RDM$BIND_SNAP_QUIET_POINT および RDB_BIND_SNAP_QUIET_POINT

論理名 RDM$BIND_SNAP_QUIET_POINT および構成パラメータ RDB_BIND_SNAP_QUIET_POINT は、スナップショット・トランザクションが静止ロックを保留し、データベースまたは After-image ジャーナル・バックアップをストールするかどうかを示します。

デフォルト値 1 は、読取り専用トランザクションが静止ロックの保留を継続することを示しています。値が 0（ゼロ）ならば、静止ポイントは読取り専用トランザクションより前に解放されます。

プロセスの変更バッファは、静止ポイント・ロックが解放される前に適切なデータベース・ファイルにフラッシュされます。

A.51 RDM$BIND_STATS_AIJ_ARBS_PER_IO および RDB_BIND_STATS_AIJ_ARBS_PER_IO

RDM$BIND_STATS_AIJ_ARBS_PER_IO 論理名および RDB_BIND_STATS_AIJ_ARBS_PER_IO 構成パラメータを使用すると、AIJ I/O あたりの AIJ 要求ブロックのデフォルト値を上書きできます。デフォルト値は 2 ブロックです。

Performance Monitor の「AIJ Analysis」画面内の構成サブメニューを使用して、このしきい値を設定することもできます。Performance Monitor の Online Analysis 機能の詳細は、2.2.16 項（2-60）を参照してください。

A.52 RDM$BIND_STATS_AIJ_BKGRD_ARB_RATIO および RDB_BIND_STATS_AIJ_BKGRD_ARB_RATIO

RDM$BIND_STATS_AIJ_BKGRD_ARB_RATIO 論理名および RDB_BIND_STATS_AIJ_BKGRD_ARB_RATIO 構成パラメータを使用すると、バックグラウンド AIJ 要求ブロックのしきい値のデフォルト値を上書きできます。デフォルト値は 50 です。

Performance Monitor の「AIJ Analysis」画面内の構成サブメニューを使用して、このしきい値を設定することもできます。Performance Monitor の Online Analysis 機能の詳細は、2.2.16 項（2-60）を参照してください。

A.53 RDM$BIND_STATS_AIJ_BLKS_PER_IO および RDB_BIND_STATS_AIJ_BLKS_PER_IO

RDM$BIND_STATS_AIJ_BLKS_PER_IO 論理名および RDB_BIND_STATS_AIJ_BLKS_PER_IO 構成パラメータを使用すると、AIJ I/O あたりのブロックのデフォルト値を上書きできます。デフォルト値は 2 です。

Performance Monitor の「AIJ Analysis」画面内の構成サブメニューを使用して、このしきい値を設定することもできます。Performance Monitor の Online Analysis 機能の詳細は、2.2.16 項（2-60）を参照してください。

A.54 RDM$BIND_STATS_AIJ_SEC_TO_EXTEND および RDB_BIND_STATS_AIJ_SEC_TO_EXTEND

RDM$BIND_STATS_AIJ_SEC_TO_EXTEND 論理名および RDB_BIND_STATS_AIJ_SEC_TO_EXTEND 構成パラメータを使用すると、AIJ を拡張する秒数のデフォルト値を上書きできます。デフォルト値は 60 です。

Performance Monitor の「AIJ Analysis」画面内の構成サブメニューを使用して、このしきい値を設定することもできます。Performance Monitor の Online Analysis 機能の詳細は、2.2.16 項（2-60）を参照してください。

A.55 RDM$BIND_STATS_BTR_FETCH_DUP_RATIO および RDB_BIND_STATS_BTR_FETCH_DUP_RATIO

RDM$BIND_STATS_BTR_FETCH_DUP_RATIO 論理名および RDB_BIND_STATS_BTR_FETCH_DUP_RATIO 構成パラメータを使用すると、B ツリーの重複フェッチのしきい値のデフォルト値を上書きできます。デフォルトのしきい値は 15 です。

Performance Monitor の「Index Analysis」画面内の構成サブメニューを使用して、このしきい値を設定することもできます。Performance Monitor の Online Analysis 機能の詳細は、2.2.16 項（2-60）を参照してください。

A.56 RDM$BIND_STATS_BTR_LEF_FETCH_RATIO および RDB_BIND_STATS_BTR_LEF_FETCH_RATIO

RDM$BIND_STATS_BTR_LEF_FETCH_RATIO 論理名および RDB_BIND_STATS_BTR_LEF_FETCH_RATIO 構成パラメータを使用すると、B ツリーのリーフ・ノード・フェッチのしきい値のデフォルト値を上書きできます。デフォルトのしきい値は 25 です。

Performance Monitor の「Index Analysis」画面内の構成サブメニューを使用して、このしきい値を設定することもできます。Performance Monitor の Online Analysis 機能の詳細は、2.2.16 項（2-60）を参照してください。

A.57 RDM$BIND_STATS_DBR_RATIO および RDB_BIND_STATS_DBR_RATIO

RDM$BIND_STATS_DBR_RATIO 論理名および RDB_BIND_STATS_DBR_RATIO 構成パラメータを使用すると、DBR を起動するしきい値のデフォルト値を上書きできます。デフォルトのしきい値は 15 です。

Performance Monitor の「RUJ Analysis」画面内の構成サブメニューを使用して、このしきい値を設定することもできます。Performance Monitor の Online Analysis 機能の詳細は、2.2.16 項（2-60）を参照してください。

A.58 RDM$BIND_STATS_ENABLED および RDB_BIND_STATS_ENABLED

論理名 RDM$BIND_STATS_ENABLED または構成パラメータ RDB_BIND_STATS_ENABLED を使用すると、プロセスに対するデータベース統計の書込みを無効にすることができます。プロセスに対するデータベース統計を無効にした場合、Performance Monitor は、そのプロセスに対して表示した各画面のフィールド内に 0 を表示します。ノード上のすべてのプロセスに対する統計の書込みを無効にするには、そのノード上の各プロセスに対して論理名 RDM$BIND_STATS_ENABLED または構成パラメータ RDB_BIND_STATS_ENABLED を定義する必要があります。デフォルトではデータベース統計の書込みはノード上の各プロセスに対して有効であり、値は 1 に設定されています。すでにチューニングされていて、Performance Monitor が提供する情報を必要としない静的かつパフォーマンスに対する要求が厳しいアプリケーションに対しては、統計を無効にすると便

利です。プロセスに対するデータベース統計の書込みを無効にするには、RDM$BIND_STATS_ENABLED 論理名または RDB_BIND_STATS_ENABLED 構成パラメータを 0 と定義します。

OpenVMS VAX OpenVMS Alpha

例 A-14（A-25）では、プロセスに対するデータベース統計を無効にする方法を示しています。

例 A-14 RDM$BIND_STATS_ENABLED 論理名の使用方法

```
$ DEFINE RDM$BIND_STATS_ENABLED 0
```

◆

データベース統計の収集が無効であるプロセスに対してデータベース統計の書込みを有効にするには、値に 1 を指定した RDM$BIND_STATS_ENABLED 論理名または RDB_BIND_STATS_ENABLED 構成パラメータを定義するか、論理名を解除するか、または構成ファイルから構成パラメータを削除します。

OpenVMS VAX OpenVMS Alpha

例 A-15（A-25）では、プロセスに対するデータベース統計を有効にする方法を示しています。

例 A-15 RDM$BIND_STATS_ENABLED 論理名の使用方法

```
$ DEFINE RDM$BIND_STATS_ENABLED 1
$
$ ! Or you can deassign the logical name
$
$ DEASSIGN RDM$BIND_STATS_ENABLED
```

◆

A.59 RDM$BIND_STATS_FULL_BACKUP_INTRVL および RDB_BIND_STATS_FULL_BACKUP_INTRVL

RDM$BIND_STATS_FULL_BACKUP_INTRVL 論理名および RDB_BIND_STATS_FULL_BACKUP_INTRVL 構成パラメータを使用すると、完全データベース・バックアップしきい値を上書きできます。デフォルトのしきい値は 6 です。

Performance Monitor の「RUJ Analysis」画面内の構成サブメニューを使用して、このしきい値を設定することもできます。Performance Monitor の Online Analysis 機能の詳細は、2.2.16 項（2-60）を参照してください。

A.60 RDM$BIND_STATS_GB_IO_SAVED_RATIO および RDB_BIND_STATS_GB_IO_SAVED_RATIO

RDM$BIND_STATS_GB_IO_SAVED_RATIO 論理名および RDB_BIND_STATS_GB_IO_SAVED_RATIO 構成パラメータを使用すると、GB IO 保存のデフォルトのしきい値を上書きできます。デフォルトのしきい値は 85 です。

Performance Monitor の「Buffer Analysis」画面内の構成サブメニューを使用して、グローバル・バッファ IO 保存しきい値を設定することもできます。Performance Monitor の Online Analysis 機能の詳細は、2.2.16 項（2-60）を参照してください。

A.61 RDM$BIND_STATS_GB_POOL_HIT_RATIO および RDB_BIND_STATS_GB_POOL_HIT_RATIO

RDM$BIND_STATS_GB_POOL_HIT_RATIO 論理名および RDB_BIND_STATS_GB_POOL_HIT_RATIO 構成パラメータを使用すると、GB プール・ヒットのデフォルトのしきい値を上書きできます。デフォルトのしきい値は 85 です。

Performance Monitor の「Buffer Analysis」画面内の構成サブメニューを使用して、グローバル・バッファ・プール・ヒットしきい値を設定することもできます。Performance Monitor の Online Analysis 機能の詳細は、2.2.16 項（2-60）を参照してください。

A.62 RDM$BIND_STATS_LB_PAGE_HIT_RATIO および RDB_BIND_STATS_LB_PAGE_HIT_RATIO

RDM$BIND_STATS_LB_PAGE_HIT_RATIO 論理名および RDB_BIND_STATS_LB_PAGE_HIT_RATIO 構成パラメータを使用すると、LB/AS ページ・ヒットのデフォルトのしきい値を上書きできます。デフォルト値は 75 です。

Performance Monitor の「Buffer Analysis」画面内の構成サブメニューを使用して、ローカル・バッファ・プール・ヒットしきい値を設定することもできます。Performance Monitor の Online Analysis 機能の詳細は、2.2.16 項（2-60）を参照してください。

A.63 RDM$BIND_STATS_MAX_HASH_QUE_LEN および RDB_BIND_STATS_MAX_HASH_QUE_LEN

RDM$BIND_STATS_MAX_HASH_QUE_LEN 論理名と RDB_BIND_STATS_MAX_HASH_QUE_LEN 構成パラメータを使用すると、ハッシュ・テーブルのキュー長のデフォルトのしきい値を上書きできます。デフォルトのしきい値は 2 行です。

Performance Monitor の「Transaction Analysis」画面内の構成サブメニューを使用して、ハッシュ・テーブルのキュー長のデフォルトのしきい値を設定することもできます。Performance Monitor の Online Analysis 機能の詳細は、2.2.16 項（2-60）を参照してください。

A.64 RDM$BIND_STATS_MAX_LOCK_STALL および RDB_BIND_STATS_MAX_LOCK_STALL

RDM$BIND_STATS_MAX_LOCK_STALL 論理名および RDB_BIND_STATS_MAX_LOCK_STALL 構成パラメータを使用すると、ロック・ストールのデフォルトのしきい値を上書きできます。デフォルトのしきい値は 2 秒です。

Performance Monitor の「Locking Analysis」画面内の構成サブメニューを使用して、このしきい値を設定することもできます。Performance Monitor の Online Analysis 機能の詳細は、2.2.16 項（2-60）を参照してください。

A.65 RDM$BIND_STATS_MAX_TX_DURATION および RDB_BIND_STATS_MAX_TX_DURATION

RDM$BIND_STATS_MAX_TX_DURATION 論理名および RDB_BIND_STATS_MAX_TX_DURATION 構成パラメータを使用すると、トランザクション継続時間のデフォルトのしきい値を上書きできます。デフォルト値は 15 です。

Performance Monitor の「Transaction Analysis」画面内の構成サブメニューを使用して、トランザクション継続時間のしきい値を設定することもできます。Performance Monitor の Online Analysis 機能の詳細は、2.2.16 項（2-60）を参照してください。

A.66 RDM$BIND_STATS_PAGES_CHECKED_RATIO および RDB_BIND_STATS_PAGES_CHECKED_RATIO

RDM$BIND_STATS_PAGES_CHECKED_RATIO 論理名および RDB_BIND_STATS_PAGES_CHECKED_RATIO 構成パラメータを使用すると、チェック済みページのデフォルトのしきい値を上書きできます。デフォルトのしきい値は 10 ページです。

Performance Monitor の「Record Analysis」画面内の構成サブメニューを使用して、このしきい値を設定することもできます。Performance Monitor の Online Analysis 機能の詳細は、2.2.16 項（2-60）を参照してください。

A.67 RDM$BIND_STATS_RECS_FETCHED_RATIO および RDB_BIND_STATS_RECS_FETCHED_RATIO

RDM$BIND_STATS_RECS_FETCHED_RATIO 論理名および RDB_BIND_STATS_RECS_FETCHED_RATIO 構成パラメータを使用すると、プリフェッチ済みレコードのデフォルトのしきい値を上書きできます。デフォルトのしきい値は 20 レコードです。

Performance Monitor の「Record Analysis」画面内の構成サブメニューを使用して、このしきい値を設定することもできます。Performance Monitor の Online Analysis 機能の詳細は、2.2.16 項（2-60）を参照してください。

A.68 RDM$BIND_STATS_RECS_STORED_RATIO および RDB_BIND_STATS_RECS_STORED_RATIO

RDM$BIND_STATS_RECS_STORED_RATIO 論理名および RDB_BIND_STATS_RECS_STORED_RATIO 構成パラメータを使用すると、格納済みレコードのデフォルトのしきい値を上書きできます。デフォルトのしきい値は 20 レコードです。

Performance Monitor の「Record Analysis」画面内の構成サブメニューを使用して、このしきい値を設定することもできます。Performance Monitor の Online Analysis 機能の詳細は、2.2.16 項（2-60）を参照してください。

A.69 RDM$BIND_STATS_RUJ_SYNC_IO_RATIO および RDB_BIND_STATS_RUJ_SYNC_IO_RATIO

RDM$BIND_STATS_GB_IO_SAVED_RATIO 論理名および RDB_BIND_STATS_GB_IO_SAVED_RATIO 構成パラメータを使用すると、同期 RUJ I/O のデフォルトのしきい値を上書きできます。デフォルトのしきい値は 10 です。

Performance Monitor の「RUJ Analysis」画面内の構成サブメニューを使用して、このしきい値を設定することもできます。Performance Monitor の Online Analysis 機能の詳細は、2.2.16 項（2-60）を参照してください。

A.70 RDM$BIND_STATS_VERB_SUCCESS_RATIO および RDB_BIND_STATS_VERB_SUCCESS_RATIO

RDM$BIND_STATS_VERB_SUCCESS_RATIO 論理名および RDB_BIND_STATS_VERB_SUCCESS_RATIO 構成パラメータを使用すると、命令の成功率のデフォルトのしきい値を上書きできます。デフォルトのしきい値は 25 です。

Performance Monitor の「Transaction Analysis」画面内の構成サブメニューを使用して、命令の成功率のしきい値を設定することもできます。Performance Monitor の Online Analysis 機能の詳細は、2.2.16 項（2-60）を参照してください。

A.71 RDM$BIND_SYSTEM_BUFFERS_ENABLED

OpenVMS Alpha

LNM$SYSTEM_TABLE 論理テーブルの論理名 RDM$BIND_SYSTEM_BUFFERS_ENABLED を使用して、システム領域グローバル・セクションが使用されるかどうかを指定できます。RDM$BIND_SYSTEM_BUFFERS_ENABLED 論理名は、システム領域グローバル・セクションを使用するすべてのノード上で定義される必要があります。論理名はデータベース・パラメータとして使用できません。

RDM$BIND_SYSTEM_BUFFERS_ENABLED 論理名には、次に示す 3 つの値があります。

0 プロセス・グローバル・セクションの使用を示します。これはデフォルトの値です。
1 可能であれば、システム領域グローバル・セクションの使用を示します。これが可能でなければプロセス・グローバル・セクションが使用されます。
2 システム領域グローバル・セクションの使用を示します。利用可能なシステム領域が十分でなければデータベースのオープン操作は失敗します。

RMU Show Users コマンドは、システム領域グローバル・セクションが有効であるかどうかを示し、グローバル・バッファ・パラメータに対するアクティブな値を示します。◆

A.72 RDM$BIND_TSN_INTERVAL および RDB_BIND_TSN_INTERVAL

RDM$BIND_TSN_INTERVAL 論理名および RDB_BIND_TSN_INTERVAL 構成パラメータを使用すると、ジャーナルへのコミットの最適化を有効にした場合、1 つのバッチとして割り当てるトランザクションの数を指定できます。

A.73 RDM$BIND_VM_SEGMENT および RDB_BIND_VM_SEGMENT

RDM$BIND_VM_SEGMENT 論理名または RDB_BIND_VM_SEGMENT 構成パラメータを使用して、メモリーの断片化を防ぎアプリケーションの仮想メモリーが不足して正しく実行されなくなることを防ぎます。断片化は仮想メモリー割当てと割当て解除が何度か行われた後で発生した非連続のバイトの集合として存在する割り当てられていない仮想メモリーです(断片と呼ばれます)。場合によっては、次の仮想メモリー要求を満たす十分な大きさのメモリーの断片がないこともあり得ます。この論理名と構成パラメータを使用して大きなブロック・サイズを予約すると、いくつかのアプリケーションで問題が発生する可能性があるので一般的には使用されません。

ユーザーが指定した仮想メモリーのバイト数を割り当てる RDMS$BIND_WORK_VM および RDB_BIND_WORK_VM とは異なり(これは、システムの使用可能なメモリー容量のみによって制限されます)、RDM$BIND_VM_SEGMENT および RDB_BIND_VM_SEGMENT はメモリーの断片化を防ぐために連続した必要なバイトを割り当てます。利用可能な仮想メモ

リー・ブロックのサイズに従って、アプリケーションは要求したバイト数より大きな仮想メモリーを受け取ります。RDM$BIND_VM_SEGMENT または RDB_BIND_VM_SEGMENT に 1 を定義するとこの機能が有効になり、0 を定義すると無効になります。

Compaq Tru64 UNIX

Compaq Tru64 UNIX 上では、パラメータを有効にするには、例 A-16（A-30）で示した行を構成ファイル内に含めてください。

例 A-16 RDB_BIND_VM_SEGMENT 構成パラメータの使用方法

```
RDB_BIND_VM_SEGMENT 1
```

◆

システム上に問題が発生しているか、その他の適切な部分、たとえば LOGIN.COM ファイルまたは構成ファイルで断片化が発生している場合は、RDM$BIND_VM_SEGMENT または RDB_BIND_VM_SEGMENT を有効にする必要があります。

A.74 RDM$BUGCHECK_DIR および RDB_BUGCHECK_DIR

RDM$BUGCHECK_DIR 論理名または RDB_BUGCHECK_DIR 構成パラメータを使用すると、バグチェック・ファイルの場所をデフォルトのディレクトリから別の場所にリダイレクトできます。これはデフォルト・ディレクトリにバグチェック・ファイル用の十分な領域がない場合に便利です。

OpenVMS VAX　OpenVMS Alpha

OpenVMS 上ではバグチェック・ディレクトリが定義された場合、OpenVMS がデフォルトで書き込むユーザーのトップレベル・ディレクトリではなく、バグチェック・ダンプ・ファイルはデバイスと RDM$BUGCHECK_DIR 論理名によってポイントされたディレクトリに書き込まれます。◆

Compaq Tru64 UNIX

Compaq Tru64 UNIX 上ではバグチェック・ディレクトリが定義された場合、Compaq Tru64 UNIX がデフォルトで書き込むデータベースが存在するディレクトリではなく、バグチェック・ダンプ・ファイルはデバイスと RDB_BUGCHECK_DIR 構成パラメータによってポイントされたディレクトリに書き込まれます。◆

バグチェック・ダンプによってデフォルト・ディレクトリ内のディスク割当てを使い切ってしまい、さらに RDM$BUGCHECK_DIR 論理名または RDB_BUGCHECK_DIR 構成パラメータに他のデバイスが指定されていない場合には、バグチェック・ダンプは KOD$TT ファイルにオーバーフローします。この場合 DBR プロセスは、KOD$TT の出力デバイスで作成されます。またバグチェック・ダンプ・ファイルはシステム・ディスク上に書き込まれ、システム・ディスクがいっぱいになると、オーバーフローは KOD$TT で終了する可能性があることにも注意してください。一般的に KOD$TT には何も書き込まれておらず、KOD$TT ファイルを検査すると、これが空であることがわかります。

OpenVMS VAX　OpenVMS Alpha

デフォルトでは、DBR バグチェック・ダンプ・ファイルは SYS$SYSTEM ディレクトリに書き込まれます。しかし、RDM$BUGCHECK_DIR 論理名を定義すると、DBR バグチェックはこの論理名で指定した新しい位置に書き込まれます。◆

他のファイルを作成する場合と同様に、バグチェック・ダンプ・ファイルを書き込むためには、ユーザーがバグチェック・ディレクトリに対して読取りと書込みのアクセス権を持っていることが必要です。

Compaq Tru64 UNIX

例 A-17（A-31）は、バグチェック・ファイルの位置のリダイレクト方法を示しています。

例 A-17 RDB_BUGCHECK_DIR 構成パラメータの使用方法

```
RDB_BUGCHECK_DIR /usr/tmp/bugcheck
```

◆

OpenVMS VAX OpenVMS Alpha

提供した論理名を NULL デバイス（NL:）に変換すると、バグチェック出力が無効になります。しかし、これはこの機能の目的に反するものであり、バグチェック出力を無効にしても、バグチェック・ダンプが発生した原因である問題は修正されません。詳細は、『Oracle Rdb7 Guide to Database Maintenance』を参照してください。◆

A.75 RDM$BUGCHECK_IGNORE_FLAGS および RDB_BUGCHECK_IGNORE_FLAGS

RDM$BUGCHECK_IGNORE_FLAGS 論理名または RDB_BUGCHECK_IGNORE_FLAGS 構成パラメータを使用すると、Oracle Rdb が作成するバグチェック・ダンプ・ファイルのサイズを削減することができます。

OpenVMS VAX OpenVMS Alpha

たとえば、OpenVMS 上のすべての Oracle Rdb ユーザーに対するロック情報とページ・ダンプのダンピングを抑制するには、次のコマンドを発行します。

```
$ DEFINE/SYSTEM RDM$BUGCHECK_IGNORE_FLAGS "LP"
```

この例では、「L」は「Locking」を表し、「P」は「Pages」を表します。◆

表 A-1（A-31）に、すべての利用可能なフラグの一覧を示します。

表 A-1 RDM$BUGCHECK_IGNORE_FLAGS および RDB_BUGCHECK_IGNORE_FLAGS

フラグ	説明
C	クライアントに固有の情報のダンプを無効にします。Oracle Rdb ではこれは、要求した BLR、要求した（REQ）ブロック、および生成したコードの情報です。この情報のダンプを無効にすることで、最適化問題、内部的に生成したコーディング問題、および領域マッピング問題など問題を診断する可能性を事実上制限します。一般的にバグチェック・ダンプのこの部分は、極端に大きくなることはありません。
G	グローバル・バッファ・データ構造のダンプを無効にします。この情報を無効にすると、グローバル・バッファに関連するバグチェックの診断がさらに困難になります。数千のグローバル・バッファを使用してオープンしているデータベースの場合、この出力は非常に大きくなる可能性があります。

表 A-1 RDM$BUGCHECK_IGNORE_FLAGS および RDB_BUGCHECK_IGNORE_FLAGS

フラグ	説明
H	ルート・ファイルの情報のダンプを無効にします。この情報は、Debug オプション付きの RMU Dump Header からの出力と同じです。この情報を無効にすると、物理記憶領域、リカバリ問題および様々な I/O サブシステムに関連した問題を診断する可能性を実質的に制限します。このセクションは、一般的にさほど大きくはありません。数百もの記憶領域を持っているデータベースは、このセクションに大量の出力を生成する可能性がありますが、一般的にダンプ出力量はさほど大きくありません。
L	ロック情報のダンプを無効にします。これはシステム用のロック・データベースのダンプです。この情報のダンプを無効にすると、ロックに関連するバグチェック診断の可能性を実質上制限します。多くのデータベースまたは大きなデータベースを持つシステムでは、この出力はきわめて大きくなる可能性があります。
P	ページ・バッファのダンプを無効にします。接続されているユーザーのバッファ内のデータベース・ページは、すべて通常どおりダンプされます。この情報を無効にすると、I/O サブシステムに関連するエラーの診断がさらに困難になる可能性があります。数百または数千のバッファが割り当てられていると、この出力はきわめて大きくなる可能性があります。

A.76 RDM$MAILBOX_CHANNEL

OpenVMS VAX OpenVMS Alpha

論理名 RDM$MAILBOX_CHANNEL には、ノード指定のデータベース・モニター・メールボックスのアドレスが含まれています。このアドレスはプロセスが適切なデータベース・モニターと通信を行うために使用します。

RDM$MAILBOX_CHANNEL 論理名の値は、永久メールボックスが最初に作成されたときにモニター・プロセスによって設定されます。この値はユーザーが設定するものではありません。

RDM$MAILBOX_CHANNEL 論理名は LNM$SYSTEM_TABLE 論理名テーブル内の LNM$PERMANENT_MAILBOX テーブル、またはテーブル群を使用して変換されます。

注意： LNM$SYSTEM_TABLE 論理名テーブル内に LNM$PERMANENT_MAILBOX テーブルが定義されていなければ、次の状態が発生します。

- データベースへ接続しようとすると、"monitor is not running" エラーが発生します。
- RMU Monitor Start コマンドがハングします。

デフォルトでは、LNM$PERMANENT_MAILBOX テーブルは、LNM$SYSTEM_TABLE 論理名テーブル内に定義されます。しかし、ユーザーやサードパーティ・アプリケーションが別の論理名テーブル（LNM$GROUP テーブルなど）内に LNM$PERMANENT_MAILBOX テーブルを再定義することがあります。これらの問題を避けるには、次に示す手順を実行してください。

1. LNM$SYSTEM_TABLE 内に LNM$PERMANENT_MAILBOX を定義

```
$ DEFINE/TABLE=LNM$PROCESS_DIRECTORY LNM$PERMANENT_MAILBOX
LNM$SYSTEM
```

2. データベース・モニターの起動

```
$ RMU/MONITOR START
```

3. アプリケーションの起動

または次の例に示すように、LNM$PERMANENT_MAILBOX テーブルを再定義するアプリケーションを変更して、LNM$SYSTEM_TABLE テーブルを含む検索リストとして、LNM$PERMANENT_MAILBOX を定義します。

```
$ DEFINE/TABLE=LNM$PROCESS_DIRECTORY LNM$PERMANENT_MAILBOX
LNM$GROUP,- _$ LNM$SYSTEM
```

◆

A.77 RDM$MONITOR および RDB_MONITOR

論理名 RDM$MONITOR と構成パラメータ RDB_MONITOR は、Oracle Rdb モニター・ログ・ファイルを配置するデバイスとディレクトリを定義します。この値には、ファイル名の指定は含めません。論理名または構成パラメータで定義するディレクトリ位置は、RMONSTART.COM コマンド・ファイルのみによってテストされ、使用されます。

注意：実際には Oracle Rdb モニターは論理名と構成パラメータを使用しません。RMONSTART.COM コマンド・ファイルだけが、論理名と構成パラメータを使用します。論理名と構成パラメータは、主にレイヤー製品で使用するために定義されています。

モニターのログ・ファイルの場所を手動で変更するには、『Oracle RMU Reference Manual』で RMU Monitor Start Output コマンドの詳細を参照してください。

OpenVMS VAX OpenVMS Alpha

RDB$MONITOR 論理名は、LNM$SYSTEM テーブルを使用して変換されます。◆

A.78 RDM$MON_USERNAME

OpenVMS VAX OpenVMS Alpha

論理名 RDM$MON_USERNAME は、起動時に、モニター・プロセスが継承するクォータのユーザー名を指定します。

システムの通常の起動では、RMU Monitor Start コマンドが実行されます。グローバル・バッファが有効であれば、PGFLQUOTA のデフォルト値、20,480（本来のプロセス・クォータ）は多数のバッファが使用されている場合作成されるモニター・プロセスに対して十分であるとはいえません。

RMU Monitor Start コマンドを使用してモニターを起動するとモニター・プロセスが使用するクォータの制限は、変更できない最小必要値、指定したユーザーのクォータ値、スタートアップを実行しているユーザーのクォータ値という 3 つの要因の大きさで決定されます。

たとえば、PGFLQUOTA プロセス・クォータの変更できない最小値が 20,480 とします。ユーザー SYSTEM のクォータ値が 40,960 で、モニターを起動したユーザーのクォータ値が 30,720 であれば、モニターは、PGFLQUOTA 値、40,960 で開始されます。

各モニター・クォータに対する変更できない最小値は、次のとおりです。

```
ASTLM              256
BIOLM              256
BYTLM           20,480
DIOLM              256
ENQLM            8,192
FILLM            1,024
PGFLQUOTA       20,480
PRCCNT              64
TQCNT               64
WSEXTENT           512
WSQUOTA            512
```

例 A-18（A-35）は、以前に説明したアルゴリズムに従って、モニター・プロセスがユーザー・アカウント GOOD_QUOTAS によって定義されているユーザ・クォータを継承することを示す RDM$MON_USERNAME 論理名の使用方法を示しています。

例 A-18 RDM$MON_USERNAME 論理名の使用方法

```
$ DEFINE RDM$MON_USERNAME GOOD_QUOTAS
```

RDM$MON_USERNAME 論理名は、グローバル・バッファを有効にした場合に、システムの起動時の RMU Monitor Start 操作で超過クォータ・エラーが発生する可能性があるため、導入されました。RDM$MON_USERNAME 論理名を使用すると、モニター・プロセスの実行に対して適切なクォータを持つ特別なアカウントを設定できます。◆

A.79 RDMS$AUTO_READY および RDB_AUTO_READY

RDMS$AUTO_READY 論理名と RDB_AUTO_READY 構成パラメータを使用すると、プロセスが CU（同時読取り）モードで論理領域のキャリーオーバー・ロックをすでに保持している場合、CR（同時更新）モードで論理領域ロックを要求しても、CU モードでロックを取得できます。

通常の環境では、プロセスは、行を最初に読み込む更新トランザクションを開始し、それを後で更新できます。この場合、Oracle Rdb は、CR（同時読取り）モードで最初にプロセスの論理領域ロックを許可し、その後ロックを CU（同時更新）モードにアップグレードします。プロセスが別の更新トランザクションを開始すると、同じ状況が発生する可能性があります。CR モードから CU モードへの推移およびその逆には、多くのロック操作が必要となる可能性があります。

プロセスに対して RDMS$AUTO_READ 論理名または RDB_AUTO_READY 構成パラメータを定義することで、プロセスに対するテーブル・レベルでの更新キャリーオーバー・ロックを有効にできます。プロセスが CR モードで論理領域ロックを要求したときに、RDMS$AUTO_READY または RDB_AUTO_READY が定義されていれば、CU モード内の論理領域でのキャリーオーバー・ロックをすでに保持している場合に、CU モードでのロックを取得できます。論理領域のキャリーオーバー・ロックを最適化すると、ほとんどの更新トランザクションに対するロックのオーバーヘッドが削減されます。

Compaq Tru64 UNIX

例 A-19（A-35）は、プロセスのテーブル・レベルで更新キャリーオーバー・ロックを有効にする方法を示しています。

例 A-19 RDB_AUTO_READY 構成パラメータの使用方法

```
RDB_AUTO_READY 1
```
◆

RDMS$AUTO_READY 論理名または RDB_AUTO_READY 構成パラメータに 1 を定義した場合、キャリーオーバー・ロックが有効ですが、それ以外の場合、テーブル・レベルで値 1 を定義しても更新キャリーオーバー・ロックは有効ではないことに注意してください。

プロセスは、データベースを更新している場合のみ、テーブルの更新キャリーオーバー・ロック最適化により実現できます。テーブル・レベルでプロセスに対する更新キャリーオーバー・ロックを有効にする利点は、プロセスが論理領域に対して CU モードでキャリーオー

バー・ロックを保持するテーブル上で新しい更新トランザクションを開始する場合に CU モードで論理領域ロックを得られることです。トランザクションの最初に CU モードで論理領域ロックを得ると、プロセスが最初にダウングレードして、後でロックをアップグレードするロックのオーバーヘッドを避けることができます。プロセスが更新を行わなければ、低い（CR）モードの論理領域ロックを取得すれば十分です。

RDMS$AUTO_READY 論理名と RDB_AUTO_READY 構成パラメータは、Performance Monitor が各トランザクションに対して多くのロック変換が行われたことを示すときに、大容量、更新集中、トランザクション処理環境で使用される必要があります。

テーブル・レベルで更新キャリーオーバー・ロックを有効にしたプロセスでは、プロセスが PROTECTED READ または PROTECTED WRITE モードでテーブルを予約する場合、またはテーブルの順次スキャンを実行する場合、並行性の問題が発生する可能性があります。

OpenVMS VAX OpenVMS Alpha

例 A-20（A-36）は、RDMS$AUTO_READY 論理名の割当てを解除することにより、プロセスに対するテーブル・レベルの更新キャリーオーバー・ロックを無効にする方法を示しています。

例 A-20 RDMS$AUTO_READY 論理名の割当てを解除することにより、テーブル・レベルで更新キャリーオーバー・ロックを無効にする方法

```
$ DEASSIGN RDMS$AUTO_READY
```

◆

A.80 RDMS$BIND_OUTLINE_FLAGS および RDB_BIND_OUTLINE_FLAGS

論理名 RDMS$BIND_OUTLINE_FLAGS と構成パラメータ RDB_BIND_OUTLINE_FLAGS を使用すると、Oracle Rdb は、クエリーのアウトラインを無視します。オプティマイザがあるクエリー用に保存したアウトラインを無視させるようにする場合、RDMS$BIND_OUTLINE_FLAGS か RDB_BIND_OUTLINE_FLAGS の値を「I」と定義します。プロセスが論理名または構成パラメータを「I」と定義した場合、オプティマイザは、クエリーを処理している間、保存された任意のアウトラインを無視します。

5.9 項（5-51）に、クエリーのアウトラインの詳細な説明があります。

A.81 RDMS$BIND_OUTLINE_MODE および RDB_BIND_OUTLINE_MODE

1 つのクエリーに対して複数のアウトラインが存在している場合、オプティマイザに使用させるアウトラインに対するアウトライン・モードの値を論理名 RDMS$BIND_OUTLINE_MODE、または構成パラメータ RDB_BIND_OUTLINE_MODE に設定します。

たとえば、あるクエリーに対して 2 つのアウトラインが保存されているとします。1 つのアウトラインはアウトライン・モードのデフォルト値である 0 を、他のアウトラインが -1 の値のアウトライン・モードを持っているとします。オプティマイザにアウトライン・モード値 0 のアウトラインを使用させる場合、RDMS$BIND_OUTLINE_MODE 論理名または RDB_BIND_OUTLINE_MODE 構成パラメータを 0（ゼロ）に設定する必要があります。オプティマイザにクエリーに対する他のアウトラインを使用させたい場合、RDMS$BIND_OUTLINE_MODE 論理名、または RDB_BIND_OUTLINE_MODE 構成パラメータを -1 に設定する必要があります。

5.9 項（5-51）に、クエリーのアウトラインの詳細な説明があります。

A.82 RDMS$BIND_PRESTART_TXN および RDB_BIND_PRESTART_TXN

RDMS$BIND_PRESTART_TXN 論理名および RDB_BIND_PRESTART_TXN 構成パラメータを使用すると、アプリケーションの外部で事前に起動したトランザクションのデフォルト設定が行えます。

この論理名および構成パラメータを使用すると、ソース・コードの変更が許可されない環境、またはソース・コードが入手できない環境では、事前に起動したトランザクションを無効にできます。

値 1 は、事前に起動したトランザクションが有効であることを、値 0 は事前に起動したトランザクションが無効であることを示しています。

OpenVMS VAX OpenVMS Alpha

次の例では、OpenVMS 上で事前に起動したトランザクションを無効にします。

```
$ DEFINE RDMS$BIND_PRESTART_TXN 0
```

◆

アプリケーションで PRESTARTED TRANSACTION 句を使用すると、RDMS$BIND_PRESTART_TXN または RDB_BIND_PRESTART_TXN に指定した値を上書きできます。

注意：Oracle Rdb は、SET TRANSACTION 文の処理で I/O を削減するデフォルトの設定である PRESTARTED TRANSACTIONS ARE ON を、アプリケーションで採用するよう推奨しています。

A.83 RDMS$BIND_QG_CPU_TIMEOUT および RDB_BIND_QG_CPU_TIMEOUT

RDMS$BIND_QG_CPU_TIMEOUT 論理名および RDB_BIND_QG_CPU_TIMEOUT 構成パラメータは、クエリーの実行の最適化に使用する CPU 時間を制限します。CPU 時間の制限に達する前にクエリーが最適化されず、実行の準備もできなければ、エラー・メッセージが返されクエリーは終了します。

デフォルトでは、クエリーのコンパイルに使用する CPU 時間には制限がありません。動的 SQL オプションは、コンパイル修飾子から継承されます。

Compaq Tru64 UNIX

例 A-21（A-38）に示す行を構成ファイルに入れると、オプティマイザが消費する CPU 経過時間を 5 秒に制限できます。

例 A-21 RDB_BIND_QG_CPU_TIMEOUT 構成パラメータの使用方法

```
RDB_BIND_QG_CPU_TIMEOUT 5
```

◆

論理名と構成パラメータは、接続時に変換され、データベースで指定したすべてのオプションを上書きします。

A.84 RDMS$BIND_QG_REC_LIMIT および RDB_BIND_QG_REC_LIMIT

RDMS$BIND_QG_REC_LIMIT 論理名または RDB_BIND_QG_REC_LIMIT 構成パラメータを使用すると、クエリーが返す行数に対して、プロセスまたはシステムの上限を設定できます。ユーザーが入力したクエリーに対し返ってきた行が、RDMS$BIND_QG_REC_LIMIT または RDB_BIND_QG_REC_LIMIT で設定した行数を超えた場合、ユーザーはエラー・メッセージを受け取り、クエリーは強制的に終了されます。これは、ユーザーが、テーブル内のすべての行、または複数のテーブルを結合したすべての行を返す一般的なクエリーを使用してシステムをオーバーロードするのを防ぎます。A.85 項（A-39）を参照すると、この目的を果たすための制限時間を設定することもできます。

OpenVMS VAX　OpenVMS Alpha

例 A-22（A-38）は、返される行の制限を設定する方法を示しています。

例 A-22 RDMS$BIND_QG_REC_LIMIT 論理名の使用方法

```
$ DEFINE RDMS$BIND_QG_REC_LIMIT 1000
```

◆

例 A-22（A-38）では、返される行数を 1000 行に制限します。指定された値はクエリーによって読み込まれるテーブル行の数とは独立しています。たとえば、結合操作が行われている間に中間行が読み出されますが、返される行の数のみに適用されます。しかし、行の制限値はクエリーの解析に必要なメタデータをフェッチするシステム・テーブル（たとえば、SQL による）のクエリーを含むすべてのデータベース・クエリーに適用されるので、極端に

小さな値の設定は避ける必要があります。行の制限値により SQL が必要なすべてのメタデータをフェッチできない場合、実行しようとしたクエリーは失敗します。

RDMS$BIND_QG_REC_LIMIT および RDB_BIND_QG_REC_LIMIT は、プロセスがデータベースに接続されたときに変換され、その値はコンパイラ修飾子を使用しているアプリケーション内で指定した制限を上書きします。

A.85 RDMS$BIND_QG_TIMEOUT および RDB_BIND_QG_TIMEOUT

RDMS$BIND_QG_TIMEOUT 論理名と RDB_BIND_QG_TIMEOUT 構成パラメータを使用すると、オプティマイザがクエリーのコンパイルに費やす時間のシステム制限を設定できます。ユーザーがクエリーを入力して RDMS$BIND_QG_TIMEOUT、または RDB_BIND_QG_TIMEOUT で設定した経過時間を超えた場合ユーザーにエラー・メッセージを送信し、クエリーを強制終了します。これはユーザーが、テーブル内のすべての行、または複数のテーブル結合内のすべての行を返す一般的なクエリーで、システムをオーバーロードするのを防ぎます。返される行数の制限を設定することもできます。A.84 項（A-38）を参照してください。

Compaq Tru64 UNIX

例 A-23（A-39）は、オプティマイザがクエリーのコンパイルに費やす時間にシステム制限を設定する方法を示しています。

例 A-23 RDB_BIND_QG_TIMEOUT 構成パラメータの使用方法

```
RDB_BIND_QG_TIMEOUT 15
```

◆

例 A-23（A-39）は、オプティマイザが費やす経過時間を 15 秒に制限します。

RDMS$BIND_QG_TIMEOUT および RDB_BIND_QG_TIMEOUT は、プロセスがデータベースに接続されたときに変換され、その値はコンパイラ修飾子を使用しているアプリケーション内で指定した制限と入れ替わります。

A.86 RDMS$BIND_SEGMENTED_STRING_BUFFER および RDB_BIND_SEGMENTED_STRING_BUFFER

RDMS$BIND_SEGMENTED_STRING_BUFFER 論理名と RDB_BIND_SEGMENTED_STRING_BUFFER 構成パラメータを使用すると、セグメント化した文字列を操作する場合の、I/O 操作のオーバーヘッドを軽減できます。

セグメント文字列のバッファ領域を増やすことで、セグメント文字列を操作するアプリケーションの効率を上げることができます。

OpenVMS VAX OpenVMS Alpha

RDML および RDBPRE プリコンパイラを使用する場合、RDMS$BIND_SEGMENTED_STRING_BUFFER 論理名に十分な大きさの値が定義されていることを確認してください。◆

デフォルトの RDB$SYSTEM 記憶領域以外の記憶領域に大きなセグメント文字列（セグメント文字列記憶マップを使用）を保存できる十分なバッファ・サイズが必要です。RDMS$BIND_SEGMENTED_STRING_BUFFER 論理名と RDB_BIND_SEGMENTED_STRING_BUFFER 構成パラメータに対する許容最小値は、セグメント文字列のセグメントの長さの合計に等しくする必要があります。たとえば、セグメントの長さの合計が 1MB であれば、1,048,576 バイトはこの論理名または構成パラメータに対して受け入れ可能な値です。

OpenVMS VAX OpenVMS Alpha

RDML および RDBPRE プリコンパイラが、セグメント文字列を保存している場合、論理名の値を指定する必要があり、すべての文字列が保存されるまで、どのテーブルがその文字列を含んでいるかは Oracle Rdb にはわかりません。可能であれば、Oracle Rdb はセグメント文字列全体をバッファリングし、STORE 文が実行されるまで保存しません。◆

セグメント文字列がバッファされたままの場合、適切な記憶領域に保存されます。（論理名、構成パラメータに対して定義された値またはデフォルト値の 10,000 バイトよりも大きいので）文字列がバッファリングされなければ、デフォルトの記憶領域に保存されず、次に示す例外メッセージが表示されます。

```
%RDB-F-IMP_EXC, facility-specific limit exceeded
-RDMS-E-SEGSTR_AREA_INC, segmented string was stored incorrectly
```

このエラーを避けるには、RDMS$BIND_SEGMENTED_STRING_BUFFER 論理名または RDB_BIND_SEGMENTED_STRING_BUFFER 構成パラメータに十分な大きさの値を設定してください。最大 500MB の値を、この論理名および構成パラメータに指定できることに注意してください。

注意：リスト（セグメント文字列）に対する SQL インタフェースには、この論理名または構成パラメータに値を定義する必要がありません。このリストがバッファに入れられる前に、SQL はリストが関連付けられている列とそれが保存されるテーブルを認識しています。しかし、大きなリストに対して、リスト全体を保持できる十分な大きさの値を持つこの論理名または構成パラメータを定義すれば、リストを保存するパフォーマンスの操作を改善できる可能性があります。

OpenVMS VAX OpenVMS Alpha

例 A-24（A-40）は、この値を 20000 バイトに変える方法を示しています。

例 A-24 RDMS$BIND_SEGMENTED_STRING_BUFFER 論理名の使用方法

```
$ DEFINE RDMS$BIND_SEGMENTED_STRING_BUFFER 20000
```

◆

A.87 RDMS$BIND_SEGMENTED_STRING_COUNT および RDB_BIND_SEGMENTED_STRING_COUNT

RDMS$BIND_SEGMENTED_STRING_COUNT 論理名と RDB_BIND_SEGMENTED_STRING_COUNT 構成パラメータは、セグメント化した文字列 ID リストでの、割当てサイズをエントリ数で指定します。この ID リストはテーブル行のセグメント文字列の実現と操作に使用します。

デフォルトでは、セグメント文字列 ID リストは、64 の整数倍のエントリ内に割り当てられており、各エントリは別のセグメント文字列を処理します。テーブルには指定された数のセグメント文字列以上が含まれているならば、以前のサイズの 2 倍のサイズを持つ別のセグメント文字列 ID リストが割り当てられます。最初のブロックには 64 エントリ、2 番目のブロックには 128 エントリが含まれ、以下同様に続きます。仮想メモリーの断片化を防ぐには、論理名または構成パラメータの値が、アプリケーションによって一般的に検索されるセグメント文字列のおよその数に近い必要があります。

RDMS$BIND_SEGMENTED_STRING_COUNT または RDB_BIND_SEGMENTED_STRING_COUNT を n と定義します。ここで、n はテーブル内のセグメント文字列の数です。デフォルト値および最小値は 64 です。

Compaq Tru64 UNIX

例 A-25（A-41）では、この構成パラメータに対する値を 100 と定義しています。

例 A-25 RDB_BIND_SEGMENTED_STRING_COUNT 構成パラメータの使用方法

```
RDB_BIND_SEGMENTED_STRING_COUNT 100
```

◆

この論理名または構成パラメータを定義することにより、次のメッセージを表示するインポート操作の失敗やプロセス・ループを防止します。

```
%SQL-F-BADBLOB, unable to import a segmented string
%RDB-F-SYS_REQUEST, error from system services request
-RDMS-F-EXQUOTA, exceeded quota
-SYSTEM-F-EXQUOTA, exceeded quota
```

この論理名または構成パラメータが複数のデータベース接続とともに使用された場合、問題が発生します。各追加接続によって、不必要な仮想メモリーの割当てが発生します。この余分な仮想メモリーはイメージが不足するまで解放されません。

オラクル社は、2 つ以上のデータベースの接続がセッションごとに実行される場合、この論理名および構成パラメータは、使用しないようにお薦めします。あるいは、この論理名と構成パラメータは、多くのセグメント文字列を持つ大きなデータベースのインポートにいつも関連するので、IMPORT 文と ATTACH 文の間で SQL 対話型ユーティリティが終了していることを確認してください。

A.88 RDMS$BIND_SEGMENTED_STRING_DBKEY_SCOPE および RDB_BIND_SEGMENTED_STRING_DBKEY_SCOPE

RDMS$BIND_SEGMENTED_STRING_DBKEY_SCOPE 論理名と RDB_BIND_SEGMENTED_STRING_DBKEY_SCOPE 構成パラメータは、プロセスが、変更したセグメント文字列の dbkey を再利用しているかどうかを示します。

セグメント文字列の最初のセグメントの dbkey は、ユーザーのデータ・レコードに保存されます。あるアプリケーションが、何か変更がなされたかどうかを決定するためにこの dbkey 値をテストすることは可能です。Oracle Rdb が同じセグメントを再利用するならば、ユーザーのアプリケーションを壊す可能性があります。

RDMS$BIND_SEGMENTED_STRING_DBKEY_SCOPE 論理名または RDB_BIND_SEGMENTED_STRING_DBKEY_SCOPE 構成パラメータが 1（真であることを示す）に設定されている場合、変更されたセグメント文字列の dbkey は、プロセスによって再利用されません。この dbkey は、データベースからプロセスが連結解除された後で再利用できます。論理名または構成パラメータが未定義である場合、または 0（偽であることを示す）が設定されている場合、変更されたセグメント文字列の dbkey は、プロセスによって再利用されます。デフォルト値は、0（偽）です。

注意：データベースの Replication Option for Rdb（公式には Data Distributor として知られている）転送が有効であれば、論理名 RDMS$BIND_SEGMENTED_STRING_DBKEY_SCOPE または構成パラメータ RDB_BIND_SEGMENTED_STRING_DBKEY_SCOPE の値は、常に 1（真）です。

OpenVMS VAX OpenVMS Alpha

例 A-26（A-42）は、プロセスがデータベースから連結解除されるまでセグメント文字列 dbkeys が再利用されないよう指示するように、この値を変更する方法を示しています。

例 A-26 RDMS$BIND_SEGMENTED_STRING_DBKEY_SCOPE 論理名の使用方法

```
$ DEFINE RDMS$BIND_SEGMENTED_STRING_DBKEY_SCOPE 1
```

◆

A.89 RDMS$BIND_SORT_WORKFILES および RDB_BIND_SORT_WORKFILES

OpenVMS VAX OpenVMS Alpha

論理名 RDMS$BIND_SORT_WORKFILES は、作業ファイルが必要な場合、OpenVMS Sort（SORT）ユーティリティがいくつの作業ファイルを使用するかを指定します。SORT のデフォルトは 2 で、最大の数は 10 です。作業ファイルは、SORTWORKn 論理名によって個別に制御できます（ここで n は 0 から 9 です）。

一時作業ファイルの場所を別のディスクに割り当てることで、SORT ユーティリティを使用する Oracle Rdb のソート操作の効率を上げることができます。これらの割当ては、SORTWORK0 から SORTWORK9 までの最大で 10 の論理名を使用することによって行えます。

通常、SORT は作業ファイルをユーザーの SYS$SCRATCH ディレクトリに配置します。デフォルトでは、SYS$SCRATCH は、SYS$LOGIN の位置と同じデバイスとディレクトリです。多くの同時ユーザーが、ソート操作が必要なクエリーを入力し、これらのユーザーは同じディスクを共有する場合、ディスク操作がボトルネックであれば、パフォーマンスに影響を与える可能性があります。ユーザーの作業ファイルを別のディスクに置くように指定すると、SORT 読取り / 書込みサイクルをオーバーラップできます。また、SYS$SCRATCH ディスク・デバイス上に十分な領域が存在しないことがありますが（たとえば、非常に大きなテーブルに対して Oracle Rdb がインデックスを構築する間）、SORTWORK0 から SORTWORK9 の論理名を使用すると、この問題を避けることができます。

ソートを効率よく行うため、作業ファイルを次のように配置します。

- 利用可能な高速デバイス
- アクティビティが最も少ないデバイス
- 利用できる領域が最も大きいデバイス
- 別のデバイス

DCL レベルでは、デバイスを指定することで任意の作業ファイルに対して異なるデバイス（例 A-27（A-43）で示す）を選択できます。

例 A-27 SORTWORKn 論理名の使用方法

```
$ DEFINE SORTWORKn   device:
```

論理名 SORTWORKn は作業ファイルです。ここで n は作業ファイルの数を示しています。デフォルトでは SORT は作業ファイルを 2 つのみ作成します。より多くの作業ファイルが指定されていなければ、RDMS$BIND_SORT_WORKFILES 論理名は、このデフォルトを使用します。したがって、Oracle Rdb ユーザーには、最大 10（SORTWORK0 から SORTWORK9）の論理名を割り当てられます。論理名 SORTWORKn を 2 通りの方法で使用できます。

- SORTWORKn を定義してデバイスを指定した場合、Oracle Rdb は隠し一時ファイルを作成します。ルート・ディレクトリは必要ありませんが、指定したデバイス上のディスク・クォータを有効にする必要があります。
- SORTWORKn を定義してデバイスとディレクトリ名を指定した場合、Oracle Rdb は参照可能な一時ファイルを作成します。ユーザーは、指定したデバイス上のディスク・クォータを有効にする必要はありません。そのかわりに、リソース識別子に対して ACL を追加して、個々のユーザーに与えることができます。その後、リソース識別子に対するディスク・クォータが使用されます。この代替手段は、すべてのアクティブ・

ユーザーに対して 1 つのディレクトリのみが必要なので、非常に多くのデータベース・ユーザーが同時にソート作業ファイルを使用している場合に最適な方法です。

例 A-28（A-44）は、最初の作業ファイルを 1DUA1 デバイスに割り当てます。

例 A-28 SORTWORK0 論理名の使用方法

```
$ DEFINE SORTWORK0   $1$DUA1:
```

ユーザーが書込み可能なディレクトリが存在する場合は、1DUA1:[USER1]SORTWORK.TMP ファイルが作成されます。ターゲット・ディスク上に書込み可能なディレクトリが存在しなければ操作は失敗します。

次の例を考えてみましょう。

- Oracle Rdb データベース CORPORATE_SALARY を構成する OpenVMS ファイルは、222DUA12、222DUA14、222DUA16、222DUA18、222DUA21 および 222DUA22 の 6 つの RA81 ディスク・デバイスに渡って分散されています。毎日の利用のピーク時間および特に週単位の給料支払い簿が計算されるとき、これらのディスクのアクティビティは非常に高くなります。
- このデータベースにアクセスしているほとんどのユーザーに対する SYS$SCRATCH 論理名は、いくつかの I/O 飽和状態のディスク 222DUA17、222DUA19 および 222DUA20 上のディレクトリに変換されます。さらに、他もこれらのディスクを使用するので、ディスクの未使用領域が常に利用できるわけではありません。
- OpenVMS Monitor ユーティリティを使用して利用可能なすべてのディスクのディスク I/O アクティビティを分析した後で、222DUA8、222DUA9、222DUA10 および 222DUA11 は最小限の I/O アクティビティを持っていると判断します。これらのディスクには、CORPORATE_SALARY データベースからの OpenVMS ファイルがまったく含まれておらず、このデータベースにアクセスする任意のユーザー（またはプログラムを実行しているユーザー）によって、SYS$SCRATCH 位置として使用されていません。
- このアプリケーションの本来の性質のため、並列 SORT 操作のハイボリュームを簡単に避ける方法はありません。

例 A-29（A-44）で示した 4 つの DCL DEFINE コマンドを入力すると、SORT 作業ファイルをユーザーの SYS$LOGIN 位置から離れたディスク上に配置します。以前に説明したように、これでソートのパフォーマンスが改善されます。

例 A-29 複数のデバイスの指定に SORTWORKn を使用する方法

```
$ DEFINE SORTWORK0 $222$DUA8:
$ DEFINE SORTWORK1 $222$DUA9:
$ DEFINE SORTWORK2 $222$DUA10:
$ DEFINE SORTWORK3 $222$DUA11:
```

◆

Compaq Tru64 UNIX

Sort ユーティリティを使用する Oracle Rdb ソート操作の効率を上げるため、一時ソート作業ファイルを異なるディスクに割り当てることで、ディスクに渡って作業ファイルを分散させることができます。環境変数 SORTWORK0 から SORTWORK255 を使用すると、これらの割当てが行えます。たとえば、次に示すコマンドを使用して、環境変数を定義できます。

```
setenv  SORTWORK0 /tmp
setenv  SORTWORK1 .
```

Sort ユーティリティが作業ファイルをいくつ使用するかを指定するには、.dbsrc 構成ファイル内で RDB_BIND_SORT_WORKFILES 構成パラメータを定義します。次の例では、Sort ユーティリティが 9 ファイルを使用することを指定する方法を示しています。

```
RDB_BIND_SORT_WORKFILES 9
```

◆

A.90 RDMS$BIND_VALIDATE_CHANGE_FIELD および RDB_BIND_VALIDATE_CHANGE_FIELD

SQL の ALTER DOMAIN 文を使用する場合、Oracle Rdb は、システム・リレーション内のメタデータを変更しますが、次の更新操作が発生したときにテーブル内に保存したデータを変換します。この方法では、通常問題は発生しません。保存したデータと新しく定義したメタデータの間で変換問題が発生し、変換したデータが読めなくなることがあります。

RDMS$BIND_VALIDATE_CHANGE_FIELD 論理名または RDB_BIND_VALIDATE_CHANGE_FIELD 構成パラメータを定義することでデータの更新時ではなく、ALTER DOMAIN 文を使用してデータ・レコードが検査され新しいメタデータ定義への変換が保証されます。

Compaq Tru64 UNIX

例 A-30（A-45）は、構成ファイル上で RDB_BIND_VALIDATE_CHANGE_FIELD 構成パラメータを定義する方法を示しています。

例 A-30 RDB_BIND_VALIDATE_CHANGE_FIELD 構成パラメータの使用方法

```
RDB_BIND_VALIDATE_CHANGE_FIELD 1
```

◆

A.91 RDMS$BIND_WORK_FILE および RDB_BIND_WORK_FILE

RDMS$BIND_WORK_FILE 論理名または RDB_BIND_WORK_FILE 構成パラメータを使用すると、Oracle Rdb がマッチング操作での使用に対して使用する一時ファイルの場所をリダイレクトできます。次に示す 3 つのオプションがあります。

OpenVMS VAX　OpenVMS Alpha

- RDMS$BIND_WORK_FILE を定義しなければ、Oracle Rdb は SYS$LOGIN で指定したデバイス上に一時ファイルを作成します。◆

- RDMS$BIND_WORK_FILE または RDB_BIND_WORK_FILE を定義して、デバイス名を指定すると、Oracle Rdb は、指定したデバイスのデフォルト・ディレクトリ内に一時ファイルを作成します。
  OpenVMS VAX OpenVMS Alpha OpenVMS の場合、これは、SYS$LOGIN 内のディスク I/O 操作の数を削減します。◆
- RDMS$BIND_WORK_FILE または RDB_BIND_WORK_FILE を定義してデバイス名とディレクトリを指定すると、Oracle Rdb は、指定したディレクトリ内に一時ファイルを作成します。こうすればファイルがディスク・クォータ割当て用のリソース識別子を使用できるようになります。

3 つのすべての場合で、Oracle Rdb はクエリーを実行しているプロセスの仮想メモリーが不足した場合にファイルを作成します。そして、クエリーが完了すればそのファイルを削除します。

OpenVMS VAX OpenVMS Alpha 例 A-31（A-46）は、一時ファイルの位置を特定のデバイスおよびディレクトリに割り当てる方法を示しています。

例 A-31 RDMS$BIND_WORK_FILE 論理名の使用方法

```
$ DEFINE RDMS$BIND_WORK_FILE WORK$DISK:[RDB.WORK]
```
◆

一時ファイルをデバイスとディレクトリに割り当て、Oracle Rdb が一時ファイルを削除する前にシステムに障害が発生すると、そのファイルを探し出して自分で削除できます。

OpenVMS VAX OpenVMS Alpha 次の例は、DCL DIRECTORY コマンドを使用して、例 A-31（A-46）で指定したファイルを検索します。

```
$ DIRECTORY WORK$DISK:[RDB.WORK]
Directory WORK$DISK:[RDB.WORK]
RDMSTTBL$WYQD02QQU4D.TMP;1     76 30-JUN-1991 19:47:04.20
Total of 1 file, 76 blocks.
```

一時ファイルの名前は、後にランダムな文字の並びが続く RDMSTTBL$ で構成されており、ファイルタイプは .tmp です。

Oracle Rdb 作業ファイルは、OpenVMS RMS ファイルなので、RMS マルチバッファおよびマルチブロック件数を設定することもできます。DCL SET RMS_DEFAULT コマンドを使用すると、/BUFFER_COUNT および /BLOCK_COUNT 修飾子に適切な値を設定することで、Oracle Rdb 一時ファイルのパフォーマンスを改善できます。◆

RDMS$BIND_WORK_FILE 論理名および RDB_BIND_WORK_FILE 構成パラメータは、RDMS$BIND_WORK_VM 論理名および RDB_BIND_WORK_VM 構成パラメータといっしょに使用されることが多くます。

A.92 RDMS$BIND_WORK_VM および RDB_BIND_WORK_VM

RDMS$BIND_WORK_VM 論理名と RDB_BIND_WORK_VM 構成パラメータを使用すると、プロセスに割り当てる仮想メモリー（VM）の総量をバイト単位で指定することで、マッチング操作に対するディスク I/O のオーバーヘッドを削減できるようになります。この割当てがすべて使用されると、新たなデータ値はディスク上の一時ファイルに書き込まれます。

OpenVMS VAX OpenVMS Alpha

RDMS$BIND_WORK_FILE が定義されていなければ、一時ファイルは、SYS$LOGIN に置かれます。◆

デフォルト値は 10,000 バイトです。指定できる値の上限は、システム上で使用可能なメモリーのサイズのみによって決まります。

Compaq Tru64 UNIX

例 A-32（A-47）は、RDB_BIND_WORK_VM 値に 25,000 バイトを定義します。

例 A-32 RDB_BIND_WORK_VM 構成パラメータの使用方法

```
RDB_BIND_WORK_VM 25000
```

◆

詳細は、3.2.1.7 項（3-20）および 8.1.3 項（8-15）を参照してください。

A.93 RDMS$DEBUG_FLAGS および RDB_DEBUG_FLAGS

RDMS$DEBUG_FLAGS 論理名と RDB_DEBUG_FLAGS 構成パラメータを使用すると、プログラムを実行するときのデータベース・アクセス・ストラテジと、これらのストラテジの見積りコストを検査することができます。詳細は、5.8.7 項（5-47）および付録 C（C-1）を参照してください。

A.94 RDMS$DEBUG_FLAGS_OUTPUT および RDB_DEBUG_FLAGS_OUTPUT

RDMS$DEBUG_FLAGS_OUTPUT 論理名と RDB_DEBUG_FLAGS_OUTPUT 構成パラメータを使用すると、プログラムを実行するときに RDMS$DEBUG_FLAGS または RDB_DEBUG_FLAGS からの出力を収集する出力ファイルに名前を付けることができます。この処理を行うためにはディレクトリに対する書込みアクセスを持つ必要があり、またこの論理名または構成パラメータに対するディスク・デバイスが存在する必要があります。

A.95 RDMS$DIAG_FLAGS および RDB_DIAG_FLAGS

RDMS$DIAG_FLAGS 論理名または RDB_DIAG_FLAGS 構成パラメータを使用すると、エラーのあるクエリーを検出できます。RDMS$DIAG_FLAGS または RDB_DIAG_FLAGS が定義されている場合、クエリー・コンパイラはソート句を含むレコード選択式（RSE）の外部のコンテキストによって総合的に定義されたソート・キーを確認します。そのような場合、クエリー・コンパイラは次に示すエラーを生成します。

```
%RDB-E-INVALID_BLR, request BLR is incorrect at offset n
-RDMS-F-SORTKEYEXT, sort key is external to RSE context
```

Compaq Tru64 UNIX

例 A-33（A-48）で示された行を構成ファイルに加えると、エラーのあるクエリーの検出を支援します。

例 A-33 RDB_DIAG_FLAGS 構成パラメータの使用方法

```
RDB_DIAG_FLAGS S
```

◆

Oracle Rdb を使用すると、同じテーブル内に古いフォーマット（連鎖）と新しいフォーマット（インデックス）のセグメント文字列を混在させることができます。連鎖セグメント文字列を新しいインデックス・フォーマットに変換することができます。たとえば、リスト・カーソルをスクロールして FETCH LAST を実行する場合、変換は望ましいものです。連鎖セグメント文字列では FETCH LAST 文を使用すると、望みのセグメントに到達する前に Oracle Rdb がすべてのセグメントを読み込む可能性があり、これは最適ではありません。インデックス・セグメント文字列の場合、FETCH LAST 文を使用すると Oracle Rdb がポインタ・セグメントと最後のデータ・セグメントのみを読み込むようになります。

OpenVMS VAX OpenVMS Alpha

RDMS$DIAG_FLAGS または RDB_DIAG_FLAGS を L と定義すると、Oracle Rdb が連鎖セグメント文字列を使用して FETCH LAST 文を実行できなくなります。

例 A-34 RDMS$DIAG_FLAGS 論理名の使用方法

```
$ DEFINE RDMS$DIAG_FLAGS L
```

◆

RDMS$DIAG_FLAGS または RDB_DIAG_FLAGS を L と定義すると、SCROLL リスト・カーソルを開こうとした場合に OPEN CURSOR 文が失敗します。

A.96 RDMS$KEEP_PREP_FILES

OpenVMS VAX OpenVMS Alpha

RDMS$KEEP_PREP_FILES 論理名を使用すると、RDBPRE プリプロセッサが中間 .mar と言語ファイルを保持するよう指定できます。RDBPRE プログラムをデバッグとしている場合で、言語ファイルを参照する必要がある場合には、これは便利です。例 A-35（A-49）に示すコマンドを使用して、これらのファイルを保持することを指定します。

例 A-35 RDMS$KEEP_PREP_FILES 論理名の使用方法

```
$ DEFINE RDMS$KEEP_PREP_FILES YES
```

◆

A.97 RDMS$RUJ および RDB_RUJ

RDMS$RUJ 論理名または RDB_RUJ 構成パラメータを使用すると、.ruj ファイルをデフォルト・ディレクトリとは異なるディスクおよびディレクトリに置くことができます。これによって、そのディレクトリの競合を削減できます。

Compaq Tru64 UNIX

構成ファイルに RDB_RUJ 構成パラメータを記述して、例 A-36（A-49）で示すように使用する位置を指定します。

例 A-36 RDB_RUJ 構成パラメータの使用方法

```
RDB_RUJ /usr/clients/journal
```

◆

詳細は、3.2.1.7 項（3-20）および 8.1.2 項（8-4）を参照してください。

A.98 RDMS$USE_OLD_CONCURRENCY および RDB_USE_OLD_CONCURRENCY

バージョン 4.2 より前のバージョンでは、SQL インタフェースには異なる分離レベルを選択する構文がありましたが、この構文は自動的に分離レベルにシリアライズ可能にアップグレードされました（程度 2 の整合性）。多くの 4GL でもデフォルトでこのモデルを選択しています。

Oracle Rdb V4.2 では完全な分離レベルのサポートが追加されており、さらにこれらの低分離レベルもサポートされています。

これらの低い分離レベルを使用しているアプリケーションが V4.2 にアップグレードした場合、これらのアプリケーションには大きな変化が発生します。より多くのロックが使用され、異なるレコードが見えるようになります。この動作は、CONSISTENCY LEVEL 2 アプリケーションでは予測されているものですが、V4.2 以降のバージョンを使用したアプリケーションではそのような環境下ではテストやチューニングは行われていません。

RDMS$USE_OLD_CONCURRENCY に 1 を定義すると、Oracle Rdb V4.2A および V5.1 で実装されたソリューションを使用したアプリケーションを V4.1 の動作に戻すことができます。これは、TPB$K_DEGREE2 を TPB$K_DEGREE3 に変換して、V4.1 の動作を維持します。TPB$K_ISOLATION_1（ISOLATION LEVEL READ COMMITTED）は、この論理名および構成パラメータによって影響を受けないことに注意してください。

オラクル社は、できる限り実用的な環境に最適な分離レベルを使用するようにアプリケーションと 4GL を調整することをお薦めします。

OpenVMS VAX OpenVMS Alpha

例 A-37（A-50）では、RDMS$USE_OLD_CONCURRENCY 論理名によって Oracle Rdb が V4.1 分離レベルの動作を使用します。RDMS$DEBUG_FLAGS 論理名を T と定義することで、Oracle Rdb が選択した分離レベルが表示されます。

例 A-37 RDMS$USE_OLD_CONCURRENCY 論理名を使用して、Oracle Rdb にバージョン 4.1 分離レベルの動作を使用させる

```
$ DEFINE RDMS$DEBUG_FLAGS "T"
$ DEFINE RDMS$USE_OLD_CONCURRENCY 1
$ SQL
SQL> ATTACH 'FILE mf_personnel';
SQL> SET TRANSACTION READ WRITE CONSISTENCY LEVEL 2;
%SQL-I-DEPR_FEATURE, Deprecated Feature: CONCURRENCY or CONSISTENCY LEVEL.  Use
ISOLATION LEVEL instead
 Compile transaction on db: X00000001
~T Transaction Parameter Block: (len=3)
0000 (00000) TPB$K_VERSION = 1
0001 (00001) TPB$K_DEGREE2 (read committed)
0002 (00002) TPB$K_WRITE (read write)
~T Concurrency option (TPB$K_DEGREE2) converted to TPB$K_DEGREE3
 Start_transaction on db: X00000001, db count=1
SQL> EXIT
 Commit_transaction on db: X00000001
 Prepare_transaction on db: X00000001
```

Oracle Rdb は、データベース用のスナップショットが無効である場合、常に読取り / 書込みトランザクション・モードを使用します。Oracle Rdb V6.0 以降では、例 A-38（A-50）に示すように、T デバッグ・フラグが使用されている場合、この表示を出力します。

例 A-38 RDMS$USE_OLD_CONCURRENCY および RDMS$DEBUG_FLAGS 論理名を使用して、スナップショットが無効である場合の読取り専用トランザクションの変換の表示

```
$ DEFINE RDMS$DEBUG_FLAGS "T"
$ DEFINE RDMS$USE_OLD_CONCURRENCY 1
$ SQL
SQL> ALTER DATABASE FILE mf_personnel
  1> SNAPSHOT IS DISABLED;
SQL> ATTACH 'FILE mf_personnel';
SQL> SET TRANSACTION READ ONLY;
 Compile transaction on db: X00000002
~T Transaction Parameter Block: (len=2)
0000 (00000) TPB$K_VERSION = 1
0001 (00001) TPB$K_READ (read only)
 Start_transaction on db: X00000002, db count=1
~T Snapshots are disabled, READ ONLY converted to READ WRITE
SQL>
```

◆

A.99 RDMS$USE_OLD_COST_MODEL および RDB_USE_OLD_COST_MODEL

RDMS$USE_OLD_COST_MODEL または RDB_USE_OLD_COST_MODEL に任意の値を定義すると、オプティマイザは作業負荷統計と記憶領域統計を使用しません。RDMS$USE_OLD_COST_MODEL または RDB_USE_OLD_COST_MODEL を使用すると、次の作業を行うことができます。

- 記憶領域統計および作業負荷統計を使用した場合と、しなかった場合でクエリーの作業負荷の最適化をテストする
- 特定のユーザー、プロセスおよびバッチ・ジョブに対する作業負荷統計および記憶領域統計の使用を選択的に無効にする

RDMS$USE_OLD_COST_MODEL 論理名または RDB_USE_OLD_COST_MODEL 構成パラメータが定義された場合、オプティマイザは、バージョン 7.0 以前に使用されていたコストとカーディナリティ機能を使用し、収集された作業負荷統計および記憶領域統計を無視します。

論理名または構成パラメータの割当てを解除してオプティマイザの実行を可能にし、作業負荷統計および記憶領域統計を再度使用してコストおよびカーディナリティの見積りを行います。

A.100 RDMS$USE_OLD_COUNT_RELATION および RDB_USE_OLD_COUNT_RELATION

以前のバージョンの Oracle Rdb では、CREATE INDEX 文は最初の行をフェッチしてテーブルが空であるかどうかを確認しました。テーブルが空であれば、Oracle Rdb はデータの収集やソート、インデックスの作成を避けることができます。この最適化は、最初の行へ高速でアクセスできる SPAM ページがある均一記憶領域では非常によく動作します。

しかし、複合フォーマット記憶領域では、テーブルに行が存在することを確認するために各ページを読み、チェックする必要があります。特に、テーブルが多くの領域に渡って分割化されている場合、CREATE INDEX は多くの I/O を実行してテーブルが空であるかどうかを判断します。

Oracle Rdb V7.0 はすべての領域に対する領域スキャンを避けるため、CREATE INDEX 内に最適化を含めます。現行のトランザクション内でテーブルが作成されたら、Oracle Rdb に対する十分な内部情報があるのでテーブルにはデータが含まれていないかどうかを知ることができます。Oracle Rdb は新規に作成されたテーブルの内部リストを維持して、この最適化をサポートします。

RESTRICTED ACCESS を使用してデータベースを接続すると、Oracle Rdb はデータベースから接続を解除するまでこの最適化を維持しています。RESTRICTED ACCESS は必須なので、現行セッションでテーブルが作成された後、Oracle Rdb はどのプロセスもそのテーブルを更新していないことを保証できます。デフォルトでは、IMPORT 文は RESTRICTED ACCESS に関して新しいデータベースに接続されているので、この最適化によって大規模な

データベースのインポートの改善を最適化できます。特に、テーブルが複合フォーマット領域にマップされていれば、ハッシュ・インデックスを使用して配置されたテーブルのインポートをより少数の I/O で行うことができます。

場合によっては、アプリケーションが多数のテーブルを作成、削除する場合、新しいテーブルの内部リストを維持することは望ましくありません。このような場合、論理名 RDMS$USE_OLD_COUNT_RELATION または構成パラメータ RDB_USE_OLD_COUNT_RELATION を定義すると、この最適化を無効にできます。論理名と構成パラメータは、存在することのみが必要であり、任意の値を定義できます。

A.101 RDMS$USE_OLD_SEGMENTED_STRING および RDB_USE_OLD_SEGMENTED_STRING

RDMS$USE_OLD_SEGMENTED_STRING 論理名または RDB_USE_OLD_SEGMENTED_STRING 構成パラメータを定義すると、デフォルトで古いフォーマット（連鎖）セグメント文字列を保持します。この論理名と構成パラメータを使用すると、すべての読取り / 書込みメディアに対して、アプリケーションが連鎖フォーマット・セグメント文字列を書き込めるようになります。一度のみの書込み記憶領域が使用されている場合、新しい（インデックス）セグメント文字列フォーマットが常に使用されることに注意してください。Oracle Rdb は、論理名と構成パラメータの値を検査しません。

OpenVMS VAX OpenVMS Alpha

この論理名を有効にするには、例 A-39（A-52）で示したように定義してください。

例 A-39 RDMS$USE_OLD_SEGMENTED_STRING 論理名の使用方法

```
$ DEFINE RDMS$USE_OLD_SEGMENTED_STRING YES
```

後で新しいセグメント文字列フォーマットを使用する場合は、例 A-40（A-52）で示すように、この論理名の割当てを解除する必要があります。

例 A-40 RDMS$USE_OLD_SEGMENTED_STRING 論理名の使用方法

```
$ DEASSIGN RDMS$USE_OLD_SEGMENTED_STRING
```

◆

新旧のセグメント文字列フォーマットの混在をサポートしています。

A.102 RDMS$USE_OLD_UPDATE_RULES および RDB_USE_OLD_UPDATE_RULES

V4.1 以降、Oracle Rdb は、デフォルトで使用可能な新しい更新規則を使用しています。新しい更新規則では、他のテーブルと直接結合しているテーブルを変更したり、行を削除したりできません。しかし、テーブルがサブクエリー内にある他のテーブルと結合している場合、このテーブルの行は変更または削除できます。SQL はテーブルの行を変更または削除で

き、同時にそのテーブルが他のテーブルと結合できる構文を持っていないので、V4.1 に導入された新しい更新規則は、SQL アプリケーションに何の影響も与えません。

OpenVMS VAX OpenVMS Alpha

しかし、結合更新クエリー（つまり、他のテーブルと結合したテーブルの行を変更または削除する更新クエリー）を含む RDO アプリケーションは、新しい更新規則の影響を受けるので、修正を加える必要があります。

Oracle Rdb は、次の結合更新クエリーに対して、エラー診断を提供します。

```
FOR E IN EMPLOYEES CROSS D IN DEGREES OVER EMPLOYEE_ID
     WITH D.DEGREE = 'MA'
   ERASE E
END_FOR
%RDMS-E-JOIN_CTX_UPD, relation EMPLOYEES is part of a join, cannot be updated
```

この更新クエリーで、従業員が MA 学位を 2 つ持っている場合、同じ従業員の行は、2 つの異なる学位の行に結合されます。したがって、Oracle Rdb は同じ列を二度削除しようとします。または、ERASE verb を使用するかわりに、前の更新クエリーで EMPLOYEES テーブルに対して MODIFY verb を使用すると、Oracle Rdb は二度以上行を変更しようとします。

V4.1 より前のバージョンでは、以前のクエリーに類似した結合更新クエリーは正しく動作するか、同じ行を二度以上削除しようとしたときにエラー診断をするか、悪くても bugcheck を生成します。いくつかの 4GL ソフトウェアおよびサード・パーティ・ソフトウェア製品も影響を受けます。

前の更新クエリーは、次に示すフォームと等価なフォームに書き換えることができます。

```
FOR E IN EMPLOYEES WITH (ANY D IN DEGREES WITH D.EMPLOYEE_ID = E.EMPLOYEE_ID
                              AND D.DEGREE = 'MA')
   ERASE E
END_FOR
```

EMPLOYEES テーブルは、もはや DEGREES テーブルに直接結合されていないので、この行は消去できます。修正された更新クエリーを使用すると、従業員の行が二度以上削除されないことを保証します。

Oracle Rdb V4.1 以降のバージョンでは、新旧両方の更新規則をサポートしています。デフォルトでは、V4.1 以降のバージョンは新しい更新規則を強制しています。Oracle Rdb が古い更新規則を継続して使用するようするには、論理名 RDMS$USE_OLD_UPDATE_RULES に 1 を定義します。

例 A-41 RDMS$USE_OLD_UPDATE_RULES 論理名の使用方法

```
$ DEFINE RDMS$USE_OLD_UPDATE_RULES 1
```

次に示す結合更新クエリーは、新しい更新規則では動作しません。また、この更新クエリーは、いくつかの給与履歴行を 2 度以上変更し、数人のマネージャに複数の昇給を行います。

```
! Give a 10% salary raise to all managers who have an MA degree.
FOR S IN SALARY_HISTORY CROSS D IN DEGREES CROSS DP IN DEPARTMENTS
   WITH S.EMPLOYEE_ID = D.EMPLOYEE_ID AND
        S.EMPLOYEE_ID = DP.MANAGER_ID AND
        S.SALARY_END MISSING          AND
        D.DEGREE = 'MA'
   MODIFY S USING S.SALARY_AMOUNT = S.SALARY_AMOUNT * 1.1
END_FOR
```

このクエリーは、次のようにサブクエリーを使用して書き換えることができます。

```
FOR S IN SALARY_HISTORY
   WITH S.SALARY_END MISSING AND
        (ANY D IN DEGREES CROSS DP IN DEPARTMENTS
         WITH S.EMPLOYEE_ID = D.EMPLOYEE_ID AND
         S.EMPLOYEE_ID = DP.MANAGER_ID AND
         D.DEGREE = 'MA')
   MODIFY S USING S.SALARY_AMOUNT = S.SALARY_AMOUNT * 1.1
END_FOR
```

この改訂クエリーは、新しい更新規則でも古い更新規則でも動作します。また、このクエリーは、昇給の資格を持つ各マネージャに対して一度の昇給が行われることを保証します。

◆

A.103 RDMS$VALIDATE_ROUTINE および RDB_VALIDATE_ROUTINE

RDMS$VALIDATE_ROUTINE 論理名または RDB_VALIDATE_ROUTINE 構成パラメータを使用すると、無効なルーチンを有効としてマークします。プロセスが RDMS$VALIDATE_ROUTINE または RDB_VALIDATE_ROUTINE に 1 を定義した場合、プロセスが読取り / 書込みトランザクション内でプロシージャをコールすると、Oracle Rdb は、無効なルーチンを有効としてマークします。

Compaq Tru64 UNIX

Compaq Tru64 UNIX の場合、例 A-42（A-54）の行を .dbsrc 構成ファイルに記述することで、RDB_VALIDATE_ROUTINE 構成パラメータに 1 を設定します。

例 A-42 RDB_VALIDATE_ROUTINE 構成パラメータの使用方法

```
RDB_VALIDATE_ROUTINE 1
```

◆

A.104 RDO$EDIT

OpenVMS VAX　OpenVMS Alpha

RDO$EDIT 論理名は、対話型 RDO クエリーを編集するために選択したシステム・エディタを示します。

現在、次の 2 つの値が使用できます。

論理値	選択したエディタ
EDT	EDIT/EDT
TPU	TPU

論理値は、システム、グループまたはプロセス・レベルで定義できます。◆

A.105 RDOINI

OpenVMS VAX OpenVMS Alpha

RDOINI 論理名は、RDO の初期化情報を格納しているファイルの名前を指定します。論理名が存在し、かつ指定したファイルが存在する場合、RDO は、このファイル内のコマンドを最初に実行してから、RDO プロンプトを表示し、入力コマンドを受け取ります。

この論理名は、LNM$FILE_DEV テーブルまたはテーブル群を使用して、変換されます。◆

A.106 RMU$EDIT

OpenVMS VAX OpenVMS Alpha

RMU$EDIT 論理名は、Performance Monitor ツール機能のノートパッドの編集に使用するシステム・エディタを示します。有効な値は、EDT、LSE、TPU です。デフォルトのエディタは EDT です。

A.107 SQL$DATABASE および SQL_DATABASE

データベースを明示的に宣言していない場合、SQL$DATABASE 論理名および SQL_DATABASE 構成パラメータは SQL が宣言するデータベースを指定します。

SQL$DATABASE または SQL_DATABASE を定義すると、データベース・アクセスに必要な文を入力する前に明示的に任意のデータベースに接続しなくても、対話型 SQL が自動的にデフォルトのデータベースを検索するために使用するデータベース・ファイル仕様が提供されます。

Compaq Tru64 UNIX

例 A-43（A-55）は、SQL_DATABASE を使用すると、明示的に接続しなくてもデータベースにアクセスできることを示しています。

例 A-43 デフォルトのデータベースを定義するために SQL_DATABASE 構成パラメータを使用する方法

```
SQL_DATABASE /usr/tmp/mf_personnel
$ SQL
SQL> SHOW TABLES
User tables in database with filename /usr/tmp/mf_personnel
```

```
    CANDIDATES
    COLLEGES
    CURRENT_INFO                A view.
    CURRENT_JOB                 A view.
    CURRENT_SALARY              A view.
    DEGREES
    DEPARTMENTS
    EMPLOYEES
    JOBS
    JOB_HISTORY
    RESUMES
    SALARY_HISTORY
    WORK_STATUS
SQL>
```

◆

SQL プリコンパイラ、モジュール・コンパイラおよび実行時システムは、明示的にエイリアス（DECLARE ALIAS）やソースファイルまたは（プリコンパイル・プログラム用に）SQL コンテキスト・ファイル内のデータベースへの接続（ATTACH）を宣言していなかった場合も、SQL$DATABASE および SQL_DATABASE を変換します。

OpenVMS VAX OpenVMS Alpha

SQL$DATABASE 論理名を使用して、リポジトリ・アクセスを指定できません。したがって、DICTIONARY IS REQUIRED 句を含めた CREATE 文または ALTER 文を実行しようとする場合、SQL$DATABASE 論理名を使用してデータベースに接続しないでください。◆

A.108 SQL$DISABLE_CONTEXT

OpenVMS VAX OpenVMS Alpha

例 A-44（A-56）に示すように、SQL$DISABLE_CONTEXT 論理名に TRUE を定義して、2 フェーズ・コミット・プロトコルを無効にできます。

例 A-44 SQL$DISABLE_CONTEXT 論理名の使用方法

```
$ DEFINE SQL$DISABLE_CONTEXT TRUE
```

この論理名はバッチ更新トランザクションを実行したい場合に、分散トランザクションを無効にするのに便利です。バッチ更新トランザクションは、Recovery-unit ジャーナル（.ruj）ファイルに書き込まないので、これらのトランザクションはロールバックできず、したがって、分散トランザクションには使用できません。詳細は、『Oracle Rdb7 Guide to Distributed Transactions』を参照してください。◆

A.109 SQL$EDIT

OpenVMS VAX OpenVMS Alpha

SQL$EDIT 論理名は、対話型 SQL クエリーを編集するために選択したシステム・エディタを示します。

現在、次の 2 つの値が使用できます。

論理値	選択したエディタ
EDT	EDIT/EDT
TPU	TPU

この論理名は、システム、グループまたはプロセス・レベルで定義できます。◆

A.110 SQLINI

OpenVMS VAX OpenVMS Alpha

SQLINI 論理名は、SQL の初期化情報を格納しているファイルの名前を指定します。論理名が存在し、かつ指定したファイルが存在する場合、SQL はこのファイル内のコマンドを最初に実行してから SQL プロンプトを表示し、入力コマンドを受け取ります。

この論理名は LNM$FILE_DEV テーブルまたはテーブル群を使用して変換されます。◆

A.111 SQL$KEEP_PREP_FILES および SQL_KEEP_PREP_FILES

SQL$KEEP_PREP_FILES 論理名または SQL_KEEP_PREP_FILES 構成パラメータを使用して、SQL プリコンパイラおよび SQL モジュール言語コンパイラが中間 .mar ファイルおよび言語ファイルを保持するよう指示します。SQL ホスト言語モジュールをデバッグする場合で、言語ファイルを参照する必要がある場合には、これは便利です。

Compaq Tru64 UNIX

Compaq Tru64 UNIX 上では、これらのファイルを保持することを指定するため、例 A-45（A-57）で示した行を構成ファイル内に記述してください。

例 A-45 SQL_KEEP_PREP_FILES 構成ファイルの使用方法

```
SQL_KEEP_PREP_FILES YES
```

◆

B

Oracle Rdb イベントベース・データ・テーブル

OpenVMS VAX OpenVMS Alpha

この付録では、Oracle Rdb PERFORMANCE および RDBEXPERT コレクション・クラス用の書式付きデータベース内のイベントベース・データ・テーブルについて説明します。現行バージョンの Oracle Rdb では、ALL クラスは、RDBEXPERT クラスと同じテーブルと列を持っているので、ALL コレクション・クラスについては説明しません。

B.1 Oracle Rdb PERFORMANCE クラス・データベース・テーブル

この項では、Oracle Rdb PERFORMANCE クラス用の書式付きデータベース内のイベントベース・データ・テーブルについて説明します。この情報を使用すると、書式付きデータベース上のデータに基づいてカスタマイズ・レポートを作成することができます。

表 B-1（B-2）は、DATABASE テーブルを示しています。

表 B-1 テーブル EPC$1_221_DATABASE の列

列名	データ型	ドメイン
COLLECTION_RECORD_ID	SMALLINT	COLLECTION_RECORD_ID_DOMAIN
IMAGE_RECORD_ID	INTEGER	IMAGE_RECORD_ID_DOMAIN
CONTEXT_NUMBER	INTEGER	CONTEXT_NUMBER_DOMAIN
TIMESTAMP_POINT	DATE VMS	
STREAM_ID	INTEGER	
CLIENT_PC	INTEGER	
DB_NAME	VARCHAR(255)	
DB_NAME_STR_ID	INTEGER	STR_ID_DOMAIN
IMAGE_NAME	VARCHAR(255)	
IMAGE_NAME_STR_ID	INTEGER	STR_ID_DOMAIN

表 B-2（B-2）は、DATABASE_ST テーブルを示しています。このテーブルは、インデックスを備えています。インデックスは、STR_ID 列で定義されており、重複を許可し、タイプはソートです。

表 B-2 テーブル EPC$1_221_DATABASE_ST の列

列名	データ型	ドメイン
STR_ID	INTEGER	STR_ID_DOMAIN
SEGMENT_NUMBER	SMALLINT	SEGMENT_NUMBER_DOMAIN
STR_SEGMENT	VARCHAR(0)	

表 B-3（B-3）は、TRANSACTION テーブルを示しています。

表 B-3 テーブル EPC$1_221_TRANSACTION の列

列名	データ型	ドメイン
COLLECTION_RECORD_ID	SMALLINT	COLLECTION_RECORD_ID_DOMAIN
IMAGE_RECORD_ID	INTEGER	IMAGE_RECORD_ID_DOMAIN
CONTEXT_NUMBER	INTEGER	CONTEXT_NUMBER_DOMAIN
TIMESTAMP_START	DATE VMS	
TIMESTAMP_END	DATE VMS	
STREAM_ID_START	INTEGER	
CLIENT_PC_START	INTEGER	
LOCK_MODE_START	SMALLINT	
TRANS_ID_START	VARCHAR(16)	
TRANS_ID_START_STR_ID	INTEGER	STR_ID_DOMAIN
GLOBAL_TID_START	VARCHAR(16)	
GLOBAL_TID_START_STR_ID	INTEGER	STR_ID_DOMAIN
DBS_READS_START	INTEGER	
DBS_WRITES_START	INTEGER	
RUJ_READS_START	INTEGER	
RUJ_WRITES_START	INTEGER	
AIJ_WRITES_START	INTEGER	
ROOT_READS_START	INTEGER	
ROOT_WRITES_START	INTEGER	
BUFFER_READS_START	INTEGER	
GET_VM_BYTES_START	INTEGER	
FREE_VM_BYTES_START	INTEGER	
LOCK_REQS_START	INTEGER	
REQ_NOT_QUEUED_START	INTEGER	
REQ_STALLS_START	INTEGER	
REQ_DEADLOCKS_START	INTEGER	

表 B-3 テーブル EPC$1_221_TRANSACTION の列（続き）

列名	データ型	ドメイン
PROM_DEADLOCKS_START	INTEGER	
LOCK_RELS_START	INTEGER	
LOCK_STALL_TIME_START	INTEGER	
BIO_START	INTEGER	
DIO_START	INTEGER	
PAGEFAULTS_START	INTEGER	
PAGEFAULT_IO_START	INTEGER	
CPU_START	INTEGER	
CURRENT_PRIO_START	SMALLINT	
VIRTUAL_SIZE_START	INTEGER	
WS_SIZE_START	INTEGER	
WS_PRIVATE_START	INTEGER	
WS_GLOBAL_START	INTEGER	
DBS_READS_END	INTEGER	
DBS_WRITES_END	INTEGER	
RUJ_READS_END	INTEGER	
RUJ_WRITES_END	INTEGER	
AIJ_WRITES_END	INTEGER	
ROOT_READS_END	INTEGER	
ROOT_WRITES_END	INTEGER	
BUFFER_READS_END	INTEGER	
GET_VM_BYTES_END	INTEGER	
FREE_VM_BYTES_END	INTEGER	
LOCK_REQS_END	INTEGER	
REQ_NOT_QUEUED_END	INTEGER	
REQ_STALLS_END	INTEGER	
REQ_DEADLOCKS_END	INTEGER	

表 B-3 テーブル EPC$1_221_TRANSACTION の列（続き）

列名	データ型	ドメイン
PROM_DEADLOCKS_END	INTEGER	
LOCK_RELS_END	INTEGER	
LOCK_STALL_TIME_END	INTEGER	
BIO_END	INTEGER	
DIO_END	INTEGER	
PAGEFAULTS_END	INTEGER	
PAGEFAULT_IO_END	INTEGER	
CPU_END	INTEGER	
CURRENT_PRIO_END	SMALLINT	
VIRTUAL_SIZE_END	INTEGER	
WS_SIZE_END	INTEGER	
WS_PRIVATE_END	INTEGER	
WS_GLOBAL_END	INTEGER	
RDB_CROSS_FAC_2	INTEGER	
RDB_CROSS_FAC_3	INTEGER	
RDB_CROSS_FAC_7	INTEGER	
RDB_CROSS_FAC_14	INTEGER	

表 B-4（B-5）は、TRANSACTION_ST テーブルを示しています。このテーブルは、インデックスを備えています。インデックスは、STR_ID 列に定義されており、重複を許可し、タイプはソートです。

表 B-4 テーブル EPC$1_221_TRANSACTION_ST の列

列名	データ型	ドメイン
STR_ID	INTEGER	STR_ID_DOMAIN
SEGMENT_NUMBER	SMALLINT	SEGMENT_NUMBER_DOMAIN
STR_SEGMENT	VARCHAR(0)	

表 B-5（B-6）は、REQUEST_ACTUAL テーブルを示しています。

表 B-5 テーブル EPC$1_221_REQUEST_ACTUAL の列

列名	データ型	ドメイン
COLLECTION_RECORD_ID	SMALLINT	COLLECTION_RECORD_ID_DOMAIN
IMAGE_RECORD_ID	INTEGER	IMAGE_RECORD_ID_DOMAIN
CONTEXT_NUMBER	INTEGER	CONTEXT_NUMBER_DOMAIN
TIMESTAMP_START	DATE VMS	
TIMESTAMP_END	DATE VMS	
DBS_READS_START	INTEGER	
DBS_WRITES_START	INTEGER	
RUJ_READS_START	INTEGER	
RUJ_WRITES_START	INTEGER	
AIJ_WRITES_START	INTEGER	
ROOT_READS_START	INTEGER	
ROOT_WRITES_START	INTEGER	
BUFFER_READS_START	INTEGER	
GET_VM_BYTES_START	INTEGER	
FREE_VM_BYTES_START	INTEGER	
LOCK_REQS_START	INTEGER	
REQ_NOT_QUEUED_START	INTEGER	
REQ_STALLS_START	INTEGER	
REQ_DEADLOCKS_START	INTEGER	
PROM_DEADLOCKS_START	INTEGER	
LOCK_RELS_START	INTEGER	
LOCK_STALL_TIME_START	INTEGER	
BIO_START	INTEGER	
DIO_START	INTEGER	
PAGEFAULTS_START	INTEGER	
PAGEFAULT_IO_START	INTEGER	
CPU_START	INTEGER	

表 B-5 テーブル EPC$1_221_REQUEST_ACTUAL の列（続き）

列名	データ型	ドメイン
CURRENT_PRIO_START	SMALLINT	
VIRTUAL_SIZE_START	INTEGER	
WS_SIZE_START	INTEGER	
WS_PRIVATE_START	INTEGER	
WS_GLOBAL_START	INTEGER	
STREAM_ID_END	INTEGER	
CLIENT_PC_END	INTEGER	
REQ_ID_END	INTEGER	
COMP_STATUS_END	INTEGER	
REQUEST_OPER_END	INTEGER	
TRANS_ID_END	VARCHAR(16)	
TRANS_ID_END_STR_ID	INTEGER	STR_ID_DOMAIN
DBS_READS_END	INTEGER	
DBS_WRITES_END	INTEGER	
RUJ_READS_END	INTEGER	
RUJ_WRITES_END	INTEGER	
AIJ_WRITES_END	INTEGER	
ROOT_READS_END	INTEGER	
ROOT_WRITES_END	INTEGER	
BUFFER_READS_END	INTEGER	
GET_VM_BYTES_END	INTEGER	
FREE_VM_BYTES_END	INTEGER	
LOCK_REQS_END	INTEGER	
REQ_NOT_QUEUED_END	INTEGER	
REQ_STALLS_END	INTEGER	
REQ_DEADLOCKS_END	INTEGER	
PROM_DEADLOCKS_END	INTEGER	

表 B-5 テーブル EPC$1_221_REQUEST_ACTUAL の列（続き）

列名	データ型	ドメイン
LOCK_RELS_END	INTEGER	
LOCK_STALL_TIME_END	INTEGER	
BIO_END	INTEGER	
DIO_END	INTEGER	
PAGEFAULTS_END	INTEGER	
PAGEFAULT_IO_END	INTEGER	
CPU_END	INTEGER	
CURRENT_PRIO_END	SMALLINT	
VIRTUAL_SIZE_END	INTEGER	
WS_SIZE_END	INTEGER	
WS_PRIVATE_END	INTEGER	
WS_GLOBAL_END	INTEGER	

表 B-6（B-8）は、REQUEST_ACTUAL_ST テーブルを示しています。このテーブルは、インデックスを備えています。インデックスは、STR_ID 列に定義されており、重複を許可し、タイプはソートです。

表 B-6 テーブル EPC$1_221_REQUEST_ACTUAL_ST の列

列名	データ型	ドメイン
STR_ID	INTEGER	STR_ID_DOMAIN
SEGMENT_NUMBER	SMALLINT	SEGMENT_NUMBER_DOMAIN
STR_SEGMENT	VARCHAR(0)	

B.2 Oracle Rdb RDBEXPERT クラス・データベース・テーブル

この項では、Oracle Rdb RDBEXPERT コレクション・クラス用の書式付きデータベース内のイベントベース・データ・テーブルについて説明します。この情報を使用すると、書式付きデータベース上のデータに基づいてカスタマイズ・レポートを作成することができます。

RDBEXPERT コレクション・クラスには、B.1 項（B-2）に示したすべてのイベントベースのデータ・テーブルと、表 B-7（B-9）および表 B-8（B-9）に示した 2 つのテーブルが含まれています。

表 B-7（B-9）は、REQUEST_BLR テーブルを示しています。

表 B-7 テーブル EPC$1_221_REQUEST_BLR の列

列名	データ型	ドメイン
COLLECTION_RECORD_ID	SMALLINT	COLLECTION_RECORD_ID_DOMAIN
IMAGE_RECORD_ID	INTEGER	IMAGE_RECORD_ID_DOMAIN
CONTEXT_NUMBER	INTEGER	CONTEXT_NUMBER_DOMAIN
TIMESTAMP_POINT	DATE VMS	
STREAM_ID	INTEGER	
TRANS_ID	INTEGER	
CLIENT_PC	INTEGER	
REQ_ID	INTEGER	
BLR	VARCHAR(127)	
BLR_STR_ID	INTEGER	STR_ID_DOMAIN

表 B-8（B-9）は、REQUEST_BLR_ST テーブルを示しています。このテーブルは、インデックスを備えています。インデックスは、STR_ID 列に定義されており、重複を許可し、タイプはソートです。

表 B-8 テーブル EPC$1_221_REQUEST_BLR_ST の列

列名	データ型	ドメイン
STR_ID	INTEGER	STR_ID_DOMAIN
SEGMENT_NUMBER	SMALLINT	SEGMENT_NUMBER_DOMAIN
STR_SEGMENT	VARCHAR(128)	

◆

C

RDMS$DEBUG_FLAGS と RDB_DEBUG_FLAGS を使用したクエリー・オプティマイザの分析

Oracle Rdb を使用すると、RDMS$DEBUG_FLAGS 論理名または RDB_DEBUG_FLAGS 構成パラメータを定義することで、クエリー実行時にクエリー・オプティマイザによって選択されたアクセス・ストラテジと、これらのストラテジの見積りコストを検査できます。この機能からの表示を分析すると、日常的なクエリーに対するインデックス指定方式を開発できるだけでなく、複雑なレコード選択式を含むクエリーに対してもインデックス指定方式を開発できます。

OpenVMS VAX OpenVMS Alpha

たとえば、RDMS$DEBUG_FLAGS 論理名を 1 つ以上のフラグとして定義できます。

```
$ DEFINE RDMS$DEBUG_FLAGS  "S"
     または
$ DEFINE RDMS$DEBUG_FLAGS  "SO"
```

◆

Compaq Tru64 UNIX

.dbsrc 構成ファイル内で、RDB_DEBUG_FLAGS 構成パラメータを 1 つ以上のフラグとして定義できます。次に例を示します。

```
RDB_DEBUG_FLAGS  "S"
    または
RDB_DEBUG_FLAGS  "SO"
```

◆

表 C-1（C-2）は、利用可能なフラグ、出力結果および参照場所を示しています。

表 C-1 RDMS$DEBUG_FLAGS 論理名および RDB_DEBUG_FLAGS 構成パラメータで使用するフラグ

フラグ	出力結果	参照先
E	トレースの実行	C.6 項（C-20）
O	クエリーの統計	C.4 項（C-15）、C.5 項（C-17）
R	ソートの統計	C.7 項（C-33）
S	クエリーのストラテジ	C.1 項（C-1）、C.5 項（C-17）、C.6 項（C-20）
Ss	非システム・クエリー用のクエリー・アウトライン定義	C.2 項（C-13）
ISs	システムが作成したクエリー用のクエリー・アウトライン定義	C.2 項（C-13）
ISsn	クエリー・アウトラインが生成される制約とトリガー名	C.2 項（C-13）
Sn	制約クエリー・ストラテジ	C.3 項（C-14）
Xt	TRACE 制御文ロギング	C.9 項（C-42）
T	トランザクション・アクティビティ	C.8 項（C-39）
\	dbkey バッファのサイズを 10 に設定	C.6 項（C-20）

RDMS$DEBUG_FLAGS および RDB_DEBUG_FLAGS に対して定義されるフラグは、大 / 小文字を区別します。したがって、RDMS$DEBUG_FLAGS 論理名または RDB_DEBUG_FLAGS 構成パラメータに 1 つ以上のフラグを定義する場合は、正しい文字で各フラグを指定しているかどうかを確認してください（大文字または小文字）。たとえば、クエリー・ストラテジを使用して制約名を表示する場合は、フラグを "Sn"（大文字の S と小文字の n）と指定していることを確認してください。フラグの大 / 小文字を間違えて使用すると、フラグが生成する出力はこの章で説明しているものと異なった結果が得られます。

RDMS$DEBUG_FLAGS_OUTPUT 論理名および RDB_DEBUG_FLAGS_OUTPUT 構成パラメータを使用すると、それぞれ RDMS$DEBUG_FLAGS および RDB_DEBUG_FLAGS_OUTPUT からの出力を収集するファイルを指定できます。これが正しく動作するには、対象ディレクトリへの書込みアクセス権が必要です。

Compaq Tru64 UNIX

たとえば、次の例で示すように、.dbsrc 構成ファイル内に RDB_DEBUG_FLAGS および RDB_DEBUG_FLAGS_OUTPUT 構成パラメータを定義することができます。

```
RDB_DEBUG_FLAGS "S"
RDB_DEBUG_FLAGS_OUTPUT debugflags_output.log
```

この場合、Oracle Rdb は、デフォルト・ディレクトリ内の debugflags_output.log ファイルに、S フラグによって生成されたすべてのストラテジ情報を書き込みます。◆

- RDMS$DEBUG_FLAGS_OUTPUT または RDB_DEBUG_FLAGS_OUTPUT を定義していなければ、デフォルトのデバイスに直接出力されます。OpenVMS 上のデフォルトのデバイスは、SYS$OUTPUT であり、CompaqTru64 UNIX 上のデフォルトのデバイスは標準出力デバイス（stdout）です。
- RDMS$DEBUG_FLAGS_OUTPUT または RDB_DEBUG_FLAGS_OUTPUT を定義するときにファイルタイプを指定しないと、ファイルが受け取るデフォルトのファイルタイプは、.lis です。
- ファイルに RDMS$DEBUG_FLAGS または RDB_DEBUG_FLAGS_OUTPUT 出力を直接行う場合、そのクエリーと結果はファイル内には表示されません。出力がデフォルト・デバイス上に表示されたら、Oracle Rdb はクエリー、RDMS$DEBUG_FLAGS または RDB_DEBUG_FLAGS_OUTPUT からの出力およびクエリーの結果を示します。

注意：データベースに接続する前に、RDMS$DEBUG_FLAGS および RDMS$DEBUG_FLAGS_OUTPUT 論理名、または RDB_DEBUG_FLAGS および RDB_DEBUG_FLAGS_OUTPUT 構成パラメータを定義する必要があります。いったん接続すると、先にデータベースの接続を解除しなければ、論理名や構成パラメータをリセットできません。

SQL の SET NOEXECUTE 文を使用して、クエリーの実行を制御できます。RDMS$DEBUG_FLAGS または RDB_DEBUG_FLAGS の S および O フラグといっしょに NOEXECUTE オプションを使用すると、実際にクエリーを実行したり、結果を表示しなくても、アクセス・ストラテジおよびクエリーのコストの見積りを検査できます。これは、有効なツールです。アクセス・ストラテジやコストを比較するために多くのクエリーをテストしても、クエリーの出力は考慮しない場合には、有効なツールです。

C.1 S フラグによるオプティマイザ・ストラテジの表示

RDMS$DEBUG_FLAGS 論理名または RDB_DEBUG_FLAGS 構成パラメータを S と定義して、クエリーを実行する場合、Oracle Rdb は、オプティマイザがクエリーの結果を生成するために使用したアクセス方法を示すフォーマットされた表示を返します。表 C-2（C-4）は、フラグ表示内に表示される表記法を示します。

表 C-2 S フラグの出力定義

出力	定義
Aggregate	COUNT、SUM、AVG、MIN、MAX または GROUP BY などの統計関数の使用状況を示します。
Aggregate-F1	EXIST または ANY を含むクエリーなど、単一値の存在の確認を示します。
Aggregate-F2	一意性の確認を示します。UNIQUE を含むクエリーの場合に使用します。
BgrOnly	バックグラウンドのみのリーフ検索タイプを示します。
Bool	キーのみブール最適化を示します。オプティマイザは、この方法を使用して行をフェッチする前に dbkey をフィルタ処理します。したがって、I/O 操作が節約されます。
Card= n	システムに関連した RDB$RELATIONS 内のフィールド RDB$CARDINALITY に保存されたテーブル・カーディナリティを示します。
Conjunct	WHERE 述語の処理を示します。また、このオプティマイザはクエリーからの式を（完全に）満足するインデックスを使用できないことを意味しています。Oracle Rdb は、いくつかの WHERE 構造体を 1 つの Conjunct の一部として処理できます。したがって、ワード Conjunct の発生する数は、クエリー内に含まれる比較の数を必ずしも反映していません。
Cross block of n entries	n エントリに対するクロス（またはネスト・ループ）結合方法を示します。
Direct lookup	使用したインデックスに重複がなく、完全一致キー述語が検索に使用され、1 つまたは 0 の dbkey を返すことを示します。
Fan= n	B ツリー・ノード・サイズ、インデックス・キー長およびインデックス・ノードの初期使用率をベースにした B ツリー・ノードの平均ファンアウト・ファクタを示します。B ツリー・ノードのファンアウトは、与えられたノードに接続されている子ノード（ブランチ）の数です。平均ファンアウトが高くなるにつれて、B ツリーが含むレベルが少なくなるので、高速シングル・キー・アクセスを促進しますが、マルチユーザー環境でのデッドロックが発生する可能性が高まります。
FFirst	高速な最初のリーフ検索タイプを示します。

表 C-2 S フラグの出力定義（続き）

出力	定義
Firstn	LIMIT TO n ROWS 句の使用を示します。この情報は、最適化処理の一部としてオプティマイザによって使用されることはありませんが、クエリーからの出力の制限として使用されます。Firstn 句の処理は、常に最後に行われるので、出力の最初の行として表示されます。
Get	データ・レコード検索に対する I/O 操作の実行を示します。
Index only retrieval	要求された情報は、インデックス内から検索されたことを示します。データ・レコード・アクセスは必要ありません。
Leaf#01	実行ツリーの最初のリーフ・ノードを示します。同じツリー内の後続のリーフ・ノードは、1 つずつ増加されます。
Match	マッチ（またはマージ・スキャン）結合方法を示します。
Max key lookup Min key lookup	インデックス全体をスキャンするかわりに直接インデックス・キー参照を示します。オプティマイザが、インデックスのみの検索を使用できる場合やクエリー内に MAX または MIN 統計機能を含む場合に使用されます。オプティマイザは、次の状況では、Max または Min 集計最適化を使用します。 ■ 昇順、降順および分割インデックスに対して使用。 ■ マルチセグメント・インデックスでは、後続のセグメントが指定されておらず、先頭セグメントが同じでなければ、オプティマイザは MAX または MIN を先頭セグメント上でのみ実行できます。 ■ クエリーが非インデックス述語を含んでいなければ、不可能です。 ■ インデックス参照中にインデックス付き述語が使用されていなければ、不可能です。
Merge block of n entries	複数のテーブルから行を返すマージ・ストラテジの使用を示します。マージ・ストラテジでは、異なるテーブルの行は連結され、ユーザーに渡されます。
NdxOnly	インデックスのみのリーフ検索型を示します。
OR index retrieval	静的 OR 最適化は、ハッシュ・インデックスまたは複数のソート・インデックスを使用して 1 つのテーブルからデータを検索することを示します。各インデックスによって選択された行は、次々に送られます。重複 dbkey は、Conjunct を適用して破棄されるので、同じ行が 2 度送られることはありません。

表 C-2 S フラグの出力定義（続き）

出力	定義
Reduce	1 つ以上の列の値をベースにした重複行の除去を示します。DISTINCT または UNION オペレータを使用したクエリーで見られます。ソートは重複除去処理の頻出部分であり、したがって、Sort 表記法はよく出力に含められます。
Retrieval by dbk of relation	要求されたデータは、直接 dbkey アクセスを使用して、検索されたことを示します。
Retrieval by index of relation	データの検索にインデックスを使用することを示します。
Retrieval sequentially of relation	テーブルまたはリレーションが順次読み込まれることを示します。
Sort	要求されたデータに出力順序が指定されていることを示します。一致結合ストラテジのためにソートが行われたか、Reduce 操作のためにソートが必要であったかのいずれかです。
Sorted	ソート順リーフ検索タイプを示します。
Temporary relation	中間結果を保存する一時テーブルの作成を示します。一時テーブルはメモリーか、ディスク上に置くことができます。RDMS$BIND_WORK_VM および RDMS$BIND_WORK_FILE 論理名または RDB_BIND_WORK_VM および RDB_BIND_WORK_FILE 構成パラメータを使用して、一時テーブルの作成に伴うディスク I/O を削減する方法は、3.2.1.7 項（3-20）を参照してください。
Zigzag	マッチ検索ストラテジのバリエーションを示します。
[l:h]	インデックス・キー範囲内の下位インデックス・キー（下位 Ikey）セグメントと上位インデックス・キー（上位 Ikey）セグメントの数を示します。l は、与えられたインデックスの範囲に対する下位 Ikey セグメントの数を表し、h は上位 Ikey セグメントの数を表します。

表 C-2 S フラグの出力定義（続き）

出力	定義
	■ 表記法 [0:0] は、インデックスの全体スキャンが行われたことを示します。 ■ 表記法 [1:1] は、インデックス・キー範囲内に 1 つの下位インデックス・キー・セグメントと 1 つの上位インデックス・キー・セグメントがあることを示しています。これは、EMP_EMPLOYEE_ID = '00164' などのクエリー内に等価な述語が存在することを示すことがあります。この場合、範囲は 1 つの値です。 ■ 表記法 [1:0] は、下位インデックス・キー・セグメントが 1 つありますが、上位インデックス・キー・セグメントはないことを示します。これは、スキャンされるインデックス・キーの範囲には、開始点はあっても、上位はないことを示します。 ■ 表記法 [1:0] は、下位インデックス・キー・セグメントはありませんが、上位インデックス・キー・セグメントは 1 つあることを示します。これは、スキャンされるインデックス・キーの範囲は、インデックスの最初から開始されますが、上位インデックス・キーの境界で終了することを示します。

[l:h...]n	オプティマイザがシングル・ソート・インデックスを使用して 2 つ以上のデータ行の範囲を指定する動的 OR 最適化を示します。l は、最初の範囲内の下位 Ikey セグメントの数を表します。h は、最初の範囲内の上位 Ikey セグメントの数を表します。... は、続きのインデックス範囲を表し、n は、範囲の総計を表します。次に例を示します。
	.
	.
	.
	`WHERE EMPLOYEE_ID IN ('00164', '00177', '00200');`
	`Leaf#01 FFirst R Card=100`
	`BgrNdx1 EMP_EMPLOYEE_ID [1:1...]3`
	.
	.
	.
	この場合、3 つの範囲は、3 人の従業員 ID の数に対応しています。

~S#0001	続く行が、クエリー番号 0001 に対するストラテジを表示することを示しています。

この項では以降、様々なアクセス・ストラテジを説明する S フラグ出力の例を示します。C.5 項（C-17）および C.6 項（C-20）に S フラグのその他の例を示します。

順次アクセス・ストラテジ

例 C-1（C-8）は、RDMS$DEBUG_FLAGS または RDB_DEBUG_FLAGS が "S" と定義された場合の簡単な順次アクセスを示します。

例 C-1 S の表示 --- 順次アクセス

```
SQL> SELECT LAST_NAME, FIRST_NAME
  1> FROM CANDIDATES
  2> ORDER BY LAST_NAME;
Sort(3)    Get(2)      Retrieval sequentially of relation CANDIDATES(1)
 LAST_NAME         FIRST_NAME
 Boswick           Fred
 Schwartz          Trixie
 Wilson            Oscar
3 rows selected
SQL>
```

次の番号は、例 C-1（C-8）の番号に対応しています。

(1) オプティマイザは、すべての行がテーブル CANDIDATES から返される必要があるので、順次検索ストラテジを選択します。

(2) 要求された行に対してフェッチされるデータ・レコードを示します。

(3) 要求された行は、ORDER BY 句で指定したようにソートされていることを示します。

クエリーのパフォーマンスを改善するために、「順次検索」という表記に特に注目してみます。関連する列に対してインデックスを定義すると、パフォーマンスを改善できる可能性があります。しかし、単にインデックスを定義するだけで、自動的にクエリーの速度が上がったり、Oracle Rdb が定義したインデックスを使用するとは考えないでください。インデックスを定義する正しい方法は、RDMS$DEBUG_FLAGS または RDB_DEBUG_FLAGS を再度使用してクエリーを実行し、「インデックス検索」という表記の観点から出力を確認することです。インデックスを追加してパフォーマンスが実際に改善されるかを判断するには、クエリーの時間を計る必要があります。場合によっては、行を順次アクセスするほうが速いため、Oracle Rdb はインデックスを無視します。また、どちらのクエリーのほうが I/O 操作が少なくて済むかを判断するため、O フラグを設定して、クエリーのコストを確認します。

さらに、ソート表記法の存在に注意してください。送出前に行をソートすると、常にクエリーの実行速度が遅くなります。適切なインデックスを定義することで、行のソートを防ぐことができるので、クエリーの実行速度をスピードアップできます。

インデックス・アクセス・ストラテジ

例 C-2（C-9）は、RDMS$DEBUG_FLAGS または RDB_DEBUG_FLAGS が "S" として定義された場合のインデックス・アクセス・ストラテジを示します。

例 C-2 S の表示 --- インデックス・アクセス

```
SQL> SELECT LAST_NAME FROM EMPLOYEES
  1> WHERE EMPLOYEE_ID = '00167';
Get (5)   Retrieval by Index  of relation EMPLOYEES (1)
  Index name EMPLOYEES_HASH (2) [1:1] (3)  Direct lookup (4)
 LAST_NAME
 Kilpatrick
1 row selected
SQL>
```

次の番号は、例 C-2（C-9）の番号に対応しています。

(1) オプティマイザは、インデックス検索は最高のストラテジであると判断します。

(2) EMPLOYEES_HASH ハッシュ・インデックスが選択されます。

(3) この場合、スキャンするインデックス範囲は、1 つの下位インデックス・キー・セグメントと 1 つの上位インデックス・キー・セグメントからなることを示します。ハッシュ・インデックスの場合、範囲は 1 つの値に制限されているので、これらの値は同じです。

(4) インデックスは、重複を許可せず、1 つまたは 0 個の dbkey を返します。

(5) インデックス自体は、クエリーで必要なすべての列を含んでいないので、オプティマイザはデータ行（データ I/O を示す）をフェッチする必要があります。

インデックスのみのアクセス・ストラテジ

例 C-3（C-9）は、RDMS$DEBUG_FLAGS または RDB_DEBUG_FLAGS が "S" として定義された場合のインデックスのみのアクセス・ストラテジを示します。

例 C-3 S の表示 --- インデックスのみのアクセス

```
SQL> SELECT LAST_NAME FROM EMPLOYEES
  1> WHERE LAST_NAME STARTING WITH 'A';
Conjunct (5)        Index only retrieval of relation EMPLOYEES (1) (4)
  Index name  EMP_LAST_NAME (2) [1:1] (3)
 LAST_NAME
 Ames
 Andriola
2 rows selected
SQL>
```

次の番号は、例 C-3（C-9）の番号に対応しています。

(1) オプティマイザは、インデックスのみの検索は最高のストラテジであると判断します。

(2) EMP_LAST_NAME インデックスが選択されます。

(3) この場合、1 つの下位インデックス・キー・セグメントと 1 つの上位インデックス・キー・セグメントで、スキャンするインデックス範囲を示します。

(4) このクエリーを満足するには、LAST_NAME 列のみが必要であり、EMP_LAST_NAME インデックスはその列をベースにしているので、インデックスそれ自体は、必要なすべてのデータを含んでいます。

(5) 返されたインデックス・データは、データが WHERE 句で指定した条件に一致しているかどうかを確認するため検査されます。

OR によるインデックス・アクセス・ストラテジ

例 C-4（C-10）は、従来の（静的）OR インデックス検索ストラテジを説明しています。例 C-4（C-10）と動的 OR 最適化を説明する例 C-5（C-11）を比較してください。

例 C-4 S の表示 ---OR インデックス検索

```
SQL> SELECT EMPLOYEE_ID, LAST_NAME
  1> FROM EMPLOYEES
  2> WHERE EMPLOYEE_ID IN ('00167', '00168');
OR index retrieval (1)
  Get     Retrieval by index of relation EMPLOYEES (3)
    Index name  EMPLOYEES_HASH [1:1]       Direct lookup (2)
  Conjunct        Get     Retrieval by index of relation EMPLOYEES (5)
    Index name  EMPLOYEES_HASH [1:1]       Direct lookup (4)
 EMPLOYEE_ID   LAST_NAME (6)
 00167         Kilpatrick
 00168         Nash
2 rows selected
SQL>
```

次の番号は、例 C-4（C-10）の番号に対応しています。

(1) クエリーが IN オペレータを含み、EMPLOYEE_ID 列上にハッシュ・インデックスが存在するので、オプティマイザは、OR インデックス検索を選択します。

(2) OR インデックス検索の最初の leg で、オプティマイザは、インデックスをスキャンせずに（Direct ルックアップ）インデックス EMPLOYEES_HASH を使用して、指定した EMPLOYEE_ID（00167）を検索します。オプティマイザは、1 つの dbkey を返します。

(3) オプティマイザは、返された dbkey を使用して、EMPLOYEES テーブルからの行を検索（Get）します。

(4) OR インデックス検索の 2 番目の leg で、オプティマイザは、インデックスをスキャンせずに（Direct ルックアップ）インデックス EMPLOYEES_HASH を使用して、次の EMPLOYEE_ID（00168）を検索します。オプティマイザは、1 つの dbkey を返します。

(5) オプティマイザは、返された dbkey を使用して、EMPLOYEES テーブルからの行を検索（Get）します。検索した行は、以前に返された行と比較されます（Conjunct）。行が異なっていれば、渡されており、行が重複していれば、破棄されます。

(6) 検索されたすべての行が、ユーザーに渡されます。

動的 OR 最適化でのインデックス・アクセス・ストラテジ

例 C-5（C-11）は、動的 OR インデックス検索ストラテジを説明しています。例 C-5（C-11）と例 C-4（C-10）を比較してください。どちらの例も同じクエリーを使用していますが、例 C-5（C-11）は、personnel データベースを使用し、例 C-4（C-10）は、mf_personnel データベースを使用しています。ハッシュ・インデックスはシングルファイル・データベース上では利用できないので、例 C-5（C-11）は 2 つのキー範囲を処理するため、ソート・インデックスを使用します。

例 C-5 S の表示 --- 動的 OR インデックス検索

```
SQL> SELECT EMPLOYEE_ID, LAST_NAME
  1> FROM EMPLOYEES
  2> WHERE EMPLOYEE_ID IN ('00167', '00168');
Leaf#01 FFirst EMPLOYEES Card=100 (1)
  BgrNdx1 EMP_EMPLOYEE_ID [1:1...]2 Fan=17 (2)
 EMPLOYEE_ID   LAST_NAME (3)
 00167         Kilpatrick
 00168         Nash
2 rows selected
SQL>
```

次の番号は、例 C-5（C-11）の番号に対応しています。

(1) オプティマイザは、100 行を含む EMPLOYEES テーブル上で高速な最初の（FFirst）タイプの動的なリーフ検索を使用します。Leaf#01 は、これが実行ツリー内の最初（で唯一）のノードであることを示します。

(2) 出力の 2 行目には、次の要素が含まれます。

- EMP_EMPLOYEE_ID は、利用可能な最高のインデックスであり、オプティマイザがデータ行の検索に使用する唯一の（BgrNdx1）インデックスです。表示はされませんが、フォアグラウンド・プロセスは、同じインデックスを使用し、実際に行を渡します。
- [1:1...]2 は、このストラテジが動的な OR 最適化であることがわかる表記法です。EMP_EMPLOYEE_ID インデックスの最初の範囲（00167）に対して、1 つの下位インデックス・キー・セグメントと 1 つの上位インデックス・キー・セグメントがあります。数値 "2" は、範囲の数、この場合は、従業員 ID00167 および 00168 を示します。

- Fan=17 は、ファンアウト・ファクタの 17、つまり、このインデックスを表す B ツリー内に平均して 17 のブランチがあることを示しています。

(3) 検索された行は、ユーザーに渡されます。

クロス検索ストラテジ

例 C-6（C-12）は、RDMS$DEBUG_FLAGS または RDB_DEBUG_FLAGS が "S" として定義された場合、クロス・ブロック検索ストラテジを使用した 2 つのテーブルの結合を説明します。

例 C-6 S の表示 --- クロス・インデックス検索

```
SQL> SELECT DISTINCT JH.EMPLOYEE_ID, J.JOB_TITLE
  1> FROM JOB_HISTORY JH, JOBS J
  2> WHERE JH.JOB_CODE = J.JOB_CODE;
Reduce  Sort (6)
Cross block of 2 entries (1)
  Cross block entry 1 (2)
    Get     Retrieval sequentially of relation JOBS (3)
  Cross block entry 2 (4)
    Conjunct        Get     Retrieval sequentially of relation JOB_HISTORY (5)
 JH.EMPLOYEE_ID   J.JOB_TITLE
 00164            Department Manager
 00164            Systems Programmer
 00165            Assistant Clerk
    .
    .
    .
 00435            Vice President
 00471            Department Manager
193 rows selected
SQL>
```

次の番号は、例 C-6（C-12）の番号に対応しています。

(1) オプティマイザは、クロス・ブロック検索ストラテジを使用して、JOBS テーブル（エントリ 1）と JOB_HISTORY テーブル（エントリ 2）内のデータにアクセスします。

(2) オプティマイザは、Cross block entry 1 を使用して処理を開始します。各クロス・ブロック・エントリは、上から下まで読み込まれます。

(3) JOBS テーブル内のデータ行は、順次アクセスされます。Get 表記法はデータ読込みを表し、オプティマイザが JOBS テーブル内の行をフェッチしていることを示します。

(4) オプティマイザは、クロス・ブロック・エントリ 2 を使用して、JOB_HISTORY テーブルを処理します。

(5) JOB_HISTORY テーブル内のデータ行は、順次アクセスされます。Get 表記法は、データ読込みを表し、オプティマイザが JOB_HISTORY テーブル内の行をフェッチしていることを示します。

(6) 各テーブル（ブロック）からの結果行は、ソートされ、重複は排除されます。

C.2 Ss、ISs および ISsn フラグ付きのオプティマイザで生成されたアウトラインの表示

Oracle Rdb オプティマイザによって生成されたクエリー・アウトラインを使用して、アウトラインを定義します。次のフラグの組合せを使用して、RDMS$DEBUG_FLAGS 論理名または RDB_DEBUG_FLAGS 構成パラメータを定義すると、クエリー・アウトライン定義が表示できるようになります。

- "Ss" フラグ -- すべての非システム・クエリーに対するクエリー・アウトライン定義を表示します。
- "ISs" フラグ --Oracle Rdb が制約とトリガー内部クエリーをコンパイルしたときにシステムが作成したクエリーに対するクエリー・アウトライン定義などを表示します。
- "ISsn" フラグ -- クエリー・アウトラインが生成された制約やトリガー（または、この両方）の名前を表示します。

OpenVMS VAX OpenVMS Alpha

たとえば、次のコマンドは、"I"（内部）フラグと "Ss"（クエリー・アウトラインのダンプ）フラグを結合しています。

```
$ DEFINE RDMS$DEBUG_FLAGS "ISs"
```

次のコマンドは、"n"（名前の表示）フラグを前の例で示したフラグに追加しています。

```
$ DEFINE RDMS$DEBUG_FLAGS "ISsn"
```

◆

"s" および "n" フラグは、小文字である必要があり、大文字の "S"（ストラテジ）フラグのすぐ後に続く必要があることに注意してください。

オプティマイザが生成するアウトラインの取込み、生成されたアウトラインの解析および生成したアウトラインをデータベースに保存する前に行う編集に対するすべての情報は、5.9.1 項（5-53）を参照してください。例 5-6（5-53）は、オプティマイザが生成したアウトラインの取込み方法を示します。

C.3 Sn フラグを使用した制約名とクエリー・ストラテジの表示方法

RDMS$DEBUG_FLAGS 論理名または RDB_DEBUG_FLAGS 構成パラメータを Sn と定義する場合、Oracle Rdb は、制約クエリー・ストラテジに従って、クエリーによって評価される制約の名前を表示します。RDMS$DEBUG_FLAGS または RDB_DEBUG_FLAGS を Sn として定義する場合、大文字の S と小文字の n を使用する必要があることに注意してください。例 C-7（C-14）は、Sn フラグの表示を示します。

例 C-7 Sn フラグの表示

```
SQL> INSERT INTO EMPLOYEES
  1> (EMPLOYEE_ID,
  2>  LAST_NAME,
  3>  FIRST_NAME,
  4>  MIDDLE_INITIAL,
  5>  ADDRESS_DATA_1,
  6>  ADDRESS_DATA_2,
  7>  CITY,
  8>  STATE,
  9>  POSTAL_CODE,
 10>  SEX,
 11>  BIRTHDAY,
 12>  STATUS_CODE)
 13> VALUES
 14> ('98765',
 15>  'Snerd',
 16>  'Mortimer',
 17>  'J',
 18>  '149 Pothole Place',
 19>  'Apartment 7',
 20>  'Nashua',
 21>  'NH',
 22>  '03060',
 23>  'M',
 24>  '07-Dec-1993',
 25>  '1');
1 row inserted
SQL> COMMIT;
~S: Constraint name  EMPLOYEES_PRIMARY_EMPLOYEE_ID (1)
Cross block of 2 entries
  Cross block entry 1
    Conjunct        Firstn  Get     Retrieval by DBK of relation EMPLOYEES
  Cross block entry 2
    Conjunct        Aggregate-F2    Get
    Retrieval by index of relation EMPLOYEES
```

```
      Index name  EMPLOYEES_HASH [1:1]       Direct lookup
~S: Constraint name  EMP_SEX_VALUES     Conjunct        Firstn  Get (2)
Retrieval by DBK of relation EMPLOYEES
~S: Constraint name  EMP_STATUS_CODE_VALUES     Conjunct        Firstn  Get (3)
Retrieval by DBK of relation EMPLOYEES
SQL>
```

次の番号は例 C-7（C-14）に対応します。

(1) EMPLOYEES テーブルの EMPLOYEE_ID 列に関する主キー制約 EMPLOYEES_PRIMARY_EMPLOYEE_ID は、従業員レコードが挿入されるごとに評価されます。

(2) EMPLOYEES テーブルの SEX 列に関するチェック制約 EMP_SEX_VALUES は、従業員レコードが挿入されるごとに評価されます。

(3) EMPLOYEES テーブルの STATUS_CODE 列に関するチェック制約 EMP_STATUS_CODE_VALUES は、従業員レコードが挿入されるごとに評価されます。

例 C-7（C-14）の要求で参照する 3 つの制約はすべてコミット時に評価されるので、COMMIT 文が発行されるまで制約はコンパイルされず、名前とストラテジは表示されません。

C.4 O フラグによる最適化統計の表示

RDMS$DEBUG_FLAGS 論理名または RDB_DEBUG_FLAGS 構成パラメータを O と定義した場合、Oracle Rdb ではクエリーの最適なソリューションを検出する前にクエリー・オプティマイザが試行して拒否したソリューションの数を示す統計が表示されます。クエリーを実行するためのコストの見積り、すなわちクエリーの結果として出力される行数の期待値も表示されます。表示例を例 C-8（C-15）に示します。

例 C-8 O フラグによる表示

```
SQL> SELECT DISTINCT JH.EMPLOYEE_ID, J.JOB_TITLE
  1> FROM JOB_HISTORY JH, JOBS J
  2> WHERE JH.JOB_CODE = J.JOB_CODE;
Solutions tried 11 (1)
Solutions blocks created 8 (2)
Created solutions pruned 5 (3)
Cost of the chosen solution      1.3184742E+03 (4)
Cardinality of chosen solution   2.4829416E+02 (5)
 JH.EMPLOYEE_ID   J.JOB_TITLE
 00164            Department Manager
 00164            Systems Programmer
 00165            Assistant Clerk
 00166            Associate Programmer
 00166            Department Manager
 00167            Associate Programmer
```

```
    .
    .
    .
 00435            Department Manager
 00435            Vice President
 00471            Department Manager
193 rows selected (6)
SQL>
```

次の番号は例 C-8（C-15）に対応します。

(1) クエリー・オプティマイザが分析した実際の検索ソリューションの数（この場合は 11）

(2) クエリー・オプティマイザが関心ありと認識した検索ソリューションの数（この場合は 8）。関心のあるソリューションは、I/O 操作が少ないものかソート順をクエリーの別の部分で使用できるものです。

(3) 他にコストの低い検索の方法が検出されたためにオプティマイザがソリューション・キューから排除したソリューションの数（この場合は 5）

(4) クエリー全体の相対的なコストの見積り（この場合は 1318 回の I/O 操作）

(5) 返される行数の見積り（この場合は 248）。オプティマイザは、多くの要因に基づいてクエリーが返すと予想される行の数を見積って、カーディナリティを決定します。たとえば、次の要因があります。

- テーブルの数と各テーブルのカーディナリティ
- クエリーの述語の選択性

(6) 実際に返される行の数。オプティマイザの見積りでは 248 行でしたが、実際の数は 193 でした。

この表示は、多くのテーブルを結合した非常に複雑なクエリーを作成する場合に便利です。ただし、このようなクエリーに伴うオーバーヘッドの一部はクエリー・オプティマイザ処理のオーバーヘッドによるものです。選択式で名前を指定した列に対してインデックスを定義したり、小規模で単純なクエリーに分割したりして、様々なクエリーを試してください。それから、S フラグと O フラグを使用してクエリーを実行し、クエリーごとにクエリー・オプティマイザが選択したアクセス・ストラテジとコストを比較します。表示される相対的なコストが最小のクエリー形式を採用します。通常、最適なソリューションではクエリーができるだけ簡素化されています。

一般に、クエリー実行コストに関する考察が必要なのは、特定のクエリーに問題がある場合のみです。一度クエリーを実行してから次にクエリーを実行するまでにデータベースは大きく変わる場合があるので、オプティマイザは実行するたびに異なるアクセス・ストラテジを選択する可能性があります。ただし、複雑な選択式を含む特殊なケースでは、ホスト言語プログラムに挿入する前にこの種の分析を実行すると有効な場合もあります。

対話型 SQL 文 SET QUERY CONFIRM を使用してオプティマイザのコスト情報にアクセスすることもできます。5.8.8 項（5-48）を参照してください。

C.5 SO フラグによる最適化ストラテジと最適化コストの表示

RDMS$DEBUG_FLAGS 論理名または RDB_DEBUG_FLAGS 構成パラメータを SO と定義した場合、Oracle Rdb ではクエリーを実行する前にクエリー・オプティマイザが試行して拒否したソリューションの数と、オプティマイザがクエリーの実行に使用した検索ストラテジ（1 つ以上）を示す統計が表示されます。

例 C-9（C-17）と例 C-11（C-18）では、同じクエリーを使用して異なるアクセス・ストラテジのメリットとデメリットを示します。

例 C-9 SO フラグによる表示

```
SQL> SELECT JOB_CODE, JOB_TITLE, MINIMUM_SALARY, MAXIMUM_SALARY
  1> FROM JOBS
  2> WHERE WAGE_CLASS= '4'
  3> AND MINIMUM_SALARY BETWEEN 10000 AND 60000;
Solutions tried 1
Solutions blocks created 1
Created solutions pruned 0
Cost of the chosen solution            9.0000000E+00  ! (9 I/OS)
Cardinality of chosen solution         5.7421874E-02  ! (0.06 rows)
Conjunct        Get     Retrieval sequentially of relation JOBS
 JOB_CODE   JOB_TITLE              MINIMUM_SALARY   MAXIMUM_SALARY
 APGM       Associate Programmer       $15,000.00       $24,000.00
 DMGR       Department Manager         $50,000.00      $100,000.00
 DSUP       Dept. Supervisor           $36,000.00       $60,000.00
 EENG       Electrical Engineer        $20,000.00       $40,000.00
 MENG       Mechanical Engineer        $20,000.00       $35,000.00
 PRGM       Programmer                 $20,000.00       $35,000.00
 SANL       Systems Analyst            $40,000.00       $60,000.00
 SPGM       Systems Programmer         $25,000.00       $50,000.00
8 rows selected
SQL>
```

このクエリーで使用する列にはインデックスが定義されていないので、オプティマイザは JOBS テーブルを順次検索して選択式で指定した行を取り出す必要があります。S フラグの出力では、唯一のアクセス・ソリューション、テーブル内の行の順次検索が作成されたことを示しています。このソリューションに伴うコスト（O フラグで表示）は、選択した行の検索に必要な I/O 操作回数の見積りです（この場合は 9）。選択されたソリューションのカーディナリティの見積り値は、0.06 行の検索です。

クエリーに ORDER BY 句がない場合、オプティマイザでは特定の行の順序が保証されません。オプティマイザはクエリーを満足するすべての行を検出します。この場合、JOBS テーブル内の行は JOB_CODE の順に格納されており、この順序は比較的安定しています。オプティマイザは、テーブル自体の行の順序と同様に JOB_CODE に関して昇順に行を返します。

注意：前述の例では、最初のクエリーと次のクエリーで行を表示する順序が異なります。順序は、どの列にインデックスが定義されているかによって変わります。列にインデックスを定義すると、オプティマイザはインデックスのノードを値に関して昇順に並べ替えます。したがって、表示されるインデックス列のデフォルトのソート順は ASCENDING です。特定の列に関して特定の順序を指定する場合は、必ずクエリーに ORDER BY 句を指定してください。管理者はオプティマイザが行の検索に使用するアクセス方法を指定できないので、デフォルトのソート順に依存することはできません。

JOBS テーブルは小規模で比較的安定しているので（すべてのジョブ・カテゴリが定義されている場合は大幅に成長する可能性が低い）、順次アクセスが最も高速なアクセス方法だと思われます。さらに、小規模なテーブルは均一記憶領域に配置し、できるだけ高速なアクセスを実現してください。複合記憶領域の順次アクセスでは、検索に関する大きなオーバーヘッドが発生します。なんらかの理由で JOBS テーブルが大きく成長した場合は、インデックスを定義して再テストすることを考えてください。実際のアクセス時間を計測し、インデックスを使用した場合に時間が短縮されたかを判断してください。一般的な規則によれば、小規模で安定なテーブルには順次アクセスが最適です。小規模なテーブルでは一意列の値が少なく、重複はありません。確信がない場合は、例 C-10（C-18）のようにインデックスを定義してテストしてください。

例 C-10（C-18）のようにインデックスを追加すると、オプティマイザはクエリーを満足する行の検索に他のアクセス・ストラテジを検討できるようになります。

例 C-10 WAGECLASS_IDX インデックスの定義

```
SQL> ATTACH 'FILENAME mf_personnel';
SQL> SET TRANSACTION READ WRITE;
SQL> CREATE INDEX WAGECLASS_IDX ON JOBS
  1> (WAGE_CLASS)
  2> TYPE IS SORTED;
SQL> COMMIT;
```

例 C-11（C-18）では例 C-9（C-17）と同じクエリーを使用していますが、例 C-10（C-18）で定義した WAGE_CLASS インデックスを使用しています。

例 C-11 SO フラグによる表示（新しい WAGECLASS_IDX インデックスの定義を使用）

```
SQL> SELECT JOB_CODE, JOB_TITLE, MINIMUM_SALARY, MAXIMUM_SALARY
  1> FROM JOBS
  2> WHERE WAGE_CLASS= '4'
  3> AND MINIMUM_SALARY BETWEEN 10000 AND 60000;
Solutions tried 2
Solutions blocks created 1
Created solutions pruned 0
Cost of the chosen solution     4.2839408E+00   !  (4 I/O)
```

```
Cardinality of chosen solution  4.5937499E-01   !  (0.5 rows)
Leaf#01 FFirst JOBS Card=15 (1)
  BgrNdx1 WAGECLASS_IDX [1:1] Fan=19 (2)
 JOB_CODE   JOB_TITLE              MINIMUM_SALARY   MAXIMUM_SALARY
 APGM       Associate Programmer         15000.00         24000.00
 DMGR       Department Manager           50000.00        100000.00
 DSUP       Dept. Supervisor             36000.00         60000.00
 EENG       Electrical Engineer          20000.00         40000.00
 MENG       Mechanical Engineer          20000.00         35000.00
 PRGM       Programmer                   20000.00         35000.00
 SANL       Systems Analyst              40000.00         60000.00
 SPGM       Systems Programmer           25000.00         50000.00
8 rows selected
SQL>
```

例 C-11（C-18）の表示では、新しいインデックスを作成するとオプティマイザが選択した検索ストラテジが変わったことを示しています。例 C-9（C-17）の順次検索のかわりに、WAGECLASS_IDX を使用できるので FFirst リーフ検索ストラテジが生成されています。次の番号は例 C-11（C-18）に対応します。

(1) この行は、15 行で構成される JOBS テーブルの高速な最初の検索タイプの動的なリーフ最適化を表します。

(2) この行は、バックグラウンド・プロセスによって WAGECLASS_IDX インデックスがオープンしたことを表します。

この場合、オプティマイザは WAGE_CLASS 列に対して定義したソート・インデックスを使用して WAGE_CLASS 列の値が 4 の列をすべて検出します。多くの行で 4 という値が繰り返し発生しているので、すなわち、この列には重複値があるので、キー値が 4 の重複ノードをすべてロックしてテストする必要があります。2 番目にテストするのは MINIMUM_SALARY 列です。この列にはインデックスはありません。WAGE_CLASS 列はインデックス付けされているので、オプティマイザはこの値が 4 の行をすべて検出し、そのグループの各行についてその行の MINIMUM_SALARY 列もクエリーで指定した範囲内にあるかどうかを調べます。

オプティマイザは、どの行がクエリーを満足するかをインデックスだけから判断するわけではありません。インデックスにはすべての情報は存在しないためです。したがって、オプティマイザはインデックスの dbkey を使用して WAGE_CLASS 列の値が 4 の行に直接アクセスし、その MINIMUM_SALARY 列が 2 番目の句 MINIMUM_SALARY BETWEEN 10000 AND 60000 で指定する値の範囲内にあるかどうかを調べます。

O フラグと S フラグを併用すると、クエリー・オプティマイザが特定のストラテジを選択する理由を解明できます。Oracle Rdb は、4 という値を使用してインデックスの検索を制限していることに注意してください。SO フラグによる表示は、WAGECLASS_IDX ソート・インデックスの 1 つの下位 Ikey とセグメントと 1 つの上位 Ikey セグメント（[1:1]）を使用し

てこの値が 4 と定義された行を検索する際にスキャンするインデックスの範囲を指定することを表します。

このクエリーに伴うコストの見積りは、WAGE_CLASS 列のみに定義したインデックスを使用した場合は 4I/O になります。Oracle Rdb は、インデックスを介して行を検索します。しかし、テーブル内の行の追加と削除が頻繁に発生する場合、Oracle Rdb はテーブルに関連する論理領域内にランダムに行を格納します。したがって、Oracle Rdb がクエリーを実行する際に選択するアクセス・ストラテジはテーブルが空だったときに選択するアクセス・ストラテジとは異なります。WAGE_CLASS 列にインデックスを定義すると、このテーブルにインデックスを定義した場合としない場合とではどちらが高性能かという前の問題に一部回答することにもなります（9I/O に対してインデックスを使用すると 4I/O）。いくつかのクエリーで時間を計測し、時間が節約されたかを判断してください。

C.6 SE フラグによる最適化・ストラテジと実行トレースの表示

オプティマイザは、動的最適化の間にストラテジを変更する場合があります。したがって、S フラグを使用してオプティマイザ・ストラテジを調べても、クエリーを実行する際の手順がすべて明らかになるわけではありません。RDMS$DEBUG_FLAGS 論理名または RDB_DEBUG_FLAGS 構成パラメータを SE と定義した場合、Oracle Rdb では次の行を含む動的最適化ストラテジの実行トレースが表示されます。

- 初期段階で 1 行
- 各バックグラウンド・インデックス終了時に 1 行
- フォアグラウンド・インデックス終了時に 1 行
- 最終段階終了時に 1 行

場合によっては、E デバッグ・フラグによる表示で前述の情報が省略されることがあります。この場合は、指定されたリーフの実行には特定のフェーズが不要であったことになります。しかし、必ず少なくとも 1 行は印刷されます。

"¥" デバッグ・フラグは、内部 dbkey バッファ・サイズを 10 dbkey と非常に小さく（テスト用に）設定しています。これで、dbkey リストを格納する一時テーブルを強制的に使用した小規模なテーブルの動的最適化をテストできます。

> **注意**： "¥" デバッグ・フラグは、テストや問題解決にのみご使用ください。このフラグを使用するとシステム・パフォーマンスが低下するので、本番向けではありません。

Compaq Tru64 UNIX

次のように .dbsrc 構成ファイル内で RDB_DEBUG_ FLAGS 構成パラメータを定義すると、実行トレースの表示に印刷されるリーフ・ノードの最大反復回数を指定できます

```
RDB_DEBUG_FLAGS "SEnnn"
```

nnn には最大反復回数が入ります。デフォルト値は 100 回です。クロス・ブロック結合ストラテジの内部ループにリーフ・ノードを含む複雑なクエリーの場合はこの制限を設定します。一部のケースでは、この種のストラテジの実行トレースでは表示が非常に長くなり、パフォーマンスが低下するとともに多くのディスク領域を使用することがあります。◆

この項の残りの部分では、S フラグと E フラグによる動的最適化（リーフ）ストラテジの詳細出力について調べます。表 C-3（C-21）に、E フラグによる表示で使用する表記を示します。

表 C-3 E フラグによる出力の定義

出力	定義
~E#000 n.0 n(n)	実行トレース（E フラグ）の行を表します。たとえば、~E#0001.01(1) の意味は次のとおりです。 ■ #0001 は、このセッションで最初のクエリー・ストラテジの実行トレースであることを示します。 ■ .01 は、このクエリー・ストラテジの実行ツリー内で最初のリーフの出力であることを示します。 ■ (1) は、このリーフの初回の反復であることを示します。
Estim Ndx:Lev/Seps/ DBKeys n:n/n/n n:n/n\n	リーフのバックグラウンド・インデックスに関するカーディナリティの再見積りを表します。バックグラウンドの詳細は、5.7.2 項（5-31）を参照してください。 ■ Estim は、見積りの行を表します。 ■ Ndx: は、統計オプティマイザが提供するバックグラウンド・インデックス番号を表します（BgrNdx1、BgrNdx2 など）。 ■ Lev は、範囲分割が発生するインデックス・ノードまたはレベル 1 のインデックス・ノードへの B ツリー降下を表します。インデックス・ノード分割は、そのインデックス・ノードがインデックス・キーの範囲全体をカバーできなくなった場合に発生します。つまり、キー範囲は複数のインデックス・ノードに分割されます。分割が発生するレベルとノードあたりの平均キー数がわかれば、オプティマイザはカーディナリティを統計オプティマイザの初期見積りよりも正確に見積ることができます。レベル 1 のインデックス・ノードに達した場合、カーディナリティはキー範囲内にあるノード内のエントリの数に重複キー・ファクタをかけた値になります。 ■ Seps は、このレベルの範囲ノード間の Ikey セパレータの数、すなわち #OfNodes - 1 を表します。 ■ DBKeys は、範囲内の dbkey の数（検索件数）を表します。

表 C-3 E フラグによる出力の定義（続き）

出力	定義

	n は、それぞれ Ndx:Lev/Seps/DBKeys 内の位置を表します。この表記については、次の例で説明します。
	`~E0005.01(1) Estim Ndx:Lev/Seps/DBKeys 1:1/1/1 2:3/1\197`
	■ 1:1/1/1
	バックグラウンド・インデックス番号は 1 で、S による表示と同様です。
	B ツリー降下はレベル 1、すなわち最低レベルで停止しています。
	キー範囲には 1 つのセパレータがあります。ただし、レベル 1 では Seps は #OfNodes - 1 でなく #OfNodes、すなわちエントリ（行 dbkey は重複キー dbkey のリストを指す dbkey）の数です。
	dbkey は 1 つです。"1" の前のスラッシュ（/）は、"1" が正確な数字である（見積りでない）ことを表します。すなわち重複キーはありません。
	■ 2:3/1\197
	バックグラウンド・インデックス番号は 2 です。
	分割レベルは 3 です。
	この分割レベルの範囲には 2 つのノードがあります（#OfNodes--1=1）。
	dbkey の見積りは 197 です。バックスラッシュ（\）が前にある場合、数値は範囲内の dbkey の見積り値（検索件数）を表します。
	次に、もう 1 つの例を示します。
	`~E0007.01(1) Estim Ndx:Lev/Seps/DBKeys 2:3/1\38 1:_52`
	■ 2:3/1\38
	バックグラウンド・インデックス番号は 2 で、S による表示と同様です。
	レベルは 3 です。
	この分割レベルの範囲には 2 つのノードがあります（#OfNodes-1=1）。
	dbkey 数の見積りは 38 です。

表 C-3 E フラグによる出力の定義（続き）

出力	定義
	■ 1:_52 バックグラウンド・インデックス番号は 1 です。 アンダースコア（_）が前にある場合、数値は統計オプティマイザの見積りによる検索レコード数です。この見積り値は、オプティマイザの動的見積りができない場合に表示されます。バックグラウンド・インデックス情報は、dbkey に関して昇順に表示されます。したがって、バックグラウンド・インデックス番号 2 の dbkey（38）はバックグラウンド・インデックス番号 1 の dbkey（52）より少ないので、この例ではまずバックグラウンド・インデックス番号 2 の情報が表示されます。
BgrNdx1	バックグラウンド・プロセスとして選択された最初のインデックスを表します。複数のインデックスがオープンしている場合は、順に番号が付けられます。
FgrNdx	フォアグラウンド・インデックスを表します。フォアグラウンド・インデックスは唯一です。
DBKeys= n	dbkey リスト（バッファまたは一時テーブル）に累積された dbkey の数を表します。インデックスをスキャンするときに、これらの dbkey はフィルタで除外されます。
Fetches=n+n	インデックス範囲をスキャンする間にインデックス・ページのフェッチのために実行された物理的な I/O 操作の回数を表します。プラス符号の左側は、インデックス・オープン段階の I/O 部分です（B ツリー降下）。プラス符号の右側は、次の I/O 部分、すなわちインデックス・スキャンまたはデータ・スキャンに関するものです。I/O 操作の回数は、特定のリーフ・スキャンの行をフェッチするためのフェッチ行からは累積されません。この行は特定のバックグラウンド・インデックスについて実行されたフェッチの回数を示します。ただし、この情報にはデータのソートや一時テーブルへの移動に必要な I/O は含まれません。
RecsOut=n	BgrNdx が完了または終了したときに、このリーフ・ノードから呼び出し側に提供されたレコードの数を表します。RecsOut カウントは、特定のリーフ・スキャンに関して新しい ~E 行が印刷されるたびに増加します。

表 C-3 E フラグによる出力の定義（続き）

出力	定義
#Bufs=n	dbkey リスト内の dbkey が指すすべての行がフェッチされる場合に読み込まれるテーブル・ページ・バッファ数の見積りを表します。#Bufs の値は、高速な最初の検索、バックグラウンドのみ、インデックスのみのリーフ検索ストラテジについて表示されます。
Fin Buf	Fin ステージがバッファから dbkey リストを取得したことを表します。
Fin Ttbl	Fin ステージが一時テーブルから dbkey リストを取得したことを表します。
Fin Seq	Fin ステージが行を順番に提供したことを表します。Fin ステージは行を順番に提供することはありませんが、格納されたカーディナリティが実際のカーディナリティよりも非常に高くなるとこの状況が発生する場合があります。
Fin ?state?	検索の終了が早すぎたために、Fin ステージが開始されなかったことを表します。
ThreLim	非常に多くの dbkey が読み込まれたことを表します。順次スキャンまたは場合によっては他のインデックス検索の方が高速です。
FtchLim	非常に多くの I/O 操作が行われたことを表します。次のインデックス・スキャンまたは順次スキャンを試行します。
Termin*	バックグラウンド・プロセスの終了を表します。
EofData	バックグラウンド・インデックス・スキャンが正常に終了したことを表します。
'CUT	バックグラウンド・プロセスまたはフォアグラウンド・プロセスが明示的な "Close Leaf" コマンドで停止したことを表します。
'ABA	他のプロセスが優先されたためにプロセスが停止（終了）したことを表します。

S フラグと E フラグによる出力を読み取る場合は、次の 3 つの規則に留意してください。

- ~S（ストラテジ）の出力行は、クエリー・コンパイルの最終段階で印刷されます。~E（実行）の出力行は、処理が完了した後、すなわち Bgr、Fgr、Fin プロセスが完了してから、クエリー実行時に印刷されます。ただし、~E の出力行は ~S の出力に相当する部分とは離れて表示されることがあります。~E の出力行では、同じクエリーの様々なリーフだけでなく様々なクエリーのリーフの組合せも可能です。~S の表記の数値は 2

つであるのに ~E の表記には 3 つの数値があるのはこのためです。3 つの数値は、~E の出力行が属するクエリー、クエリー内のリーフのインスタンス、特定のリーフの反復を表します。

- ~E の出力行は、クエリー出力の最終行（最後に提供される行）のすぐ上に印刷されます。
- RecsOut の行は、その時点までに提供された行の総数が表示されます。すなわち、その行はフォアグラウンド・プロセスか Fin プロセスで提供されたことを意味します。ただし、バックグラウンド・プロセス終了時に提供される量は Bgr 行に表示されます。

この項の残りの部分では、S フラグと E フラグの設定によるデバッグ・フラグの出力例を示します。

実行トレースを伴う高速な最初のリーフ検索ストラテジ

例 C-12（C-25）に、高速な最初の（FFirst）リーフ検索のストラテジを示します。

例 C-12 SE フラグによる表示 ---FFirst 検索

```
SQL> SELECT LAST_NAME, EMPLOYEE_ID FROM EMPLOYEES
  1>  WHERE EMPLOYEE_ID > '00200';
~S#0004 (1)
Leaf#01 FFirst EMPLOYEES Card=100 (2)
  BgrNdx1 EMP_EMPLOYEE_ID [1:0] Fan=17 (3)
 LAST_NAME        EMPLOYEE_ID
 Clinton          00201 (4)
 Harrington       00202
 Gaudet           00203
.
.
.
 Dement           00405
 Mistretta        00415
 Ames             00416
 Blount           00418
 MacDonald        00435
~E#0004.01(1) BgrNdx1 EofData  DBKeys=63  Fetches=0+0  RecsOut=63 #Bufs=24 (5)
~E#0004.01(1) FgrNdx  FFirst   DBKeys=63  Fetches=0+23 RecsOut=63`ABA (6)
~E#0004.01(1) Fin     Buf      DBKeys=63  Fetches=0+0  RecsOut=63 (7)
 Herbener         00471 (8)
63 rows selected
SQL>
```

次の番号は例 C-12（C-25）に対応します。

(1) これは、このセッションの 4 番目のクエリーです。

(2) これはこのクエリーの実行ツリー内で最初の（そして唯一の）リーフ・ノードです。オプティマイザは、カーディナリティが 100 行の EMPLOYEES テーブルへのアクセスに高速な最初のリーフ検索タイプを選択しました。

(3) バックグラウンド・プロセスでは 1 つのインデックス（BgrNdx1）、EMP_EMPLOYEE_ID を使用します。これは、オプティマイザが最適なインデックスと判断したものです。これは FFirst 検索なので、バックグラウンド・プロセスとフォアグラウンド・プロセスの両方で EMP_EMPLOYEE_ID を使用します。
クエリーにはブール演算子の「より大きい」（>）が指定されているので、インデックス・キー値の下限はありますが上限はありません。fanout ファクタは、インデックス・ノードあたり平均 17 ブランチです。

(4) フォアグラウンド・プロセスによってただちに行の表示が開始され、最初の実行トレース行が表示されるまでに 3 行が表示されます。~E の出力行は、常にクエリー出力の最終行（最後に提供される行）の直前に印刷されます。

(5) この実行トレース行には、次の情報が表示されます。

- ~E#0004.01(1)
  この実行トレース行は、現在のセッションの 4 番目のクエリーの、実行ツリー内の最初のリーフの、このリーフの最初の反復を対象とするものです。
- BgrNdx1 EofData DBKeys=63 Fetches=0+0 RecsOut=63 #Bufs=24
  バックグラウンド・プロセスがインデックス・スキャンを終了したことを表します（EofData）。DBKeys=63 は、インデックスをスキャンする間にバッファ内に 63 個の dbkey が収集されたことを表します。Fetches の表記は、このリーフのスキャンには I/O 操作が不要だったことを表します。#Bufs の表記は、dbkey リスト内の dbkey が指すすべての行がフェッチされた場合に読み込まれるテーブル・ページ・バッファの数の見積り値を表します。

(6) ~E#0004.01(1) FgrNdx FFirst DBKeys=63 Fetches=0+23 RecsOut=63`ABA
FgrNdx FFirst の表記は、バックグラウンド・インデックスはフォアグラウンド・プロセスが提供した dbkey のソースであることを表します。RecsOut の合計 63 個のはバックグラウンド・プロセスが提供した 63 個から増えていないので、フォアグラウンド・プロセス自体は dbkey をフェッチしていません。Fetches の表記は、データ・レコードのフェッチに 23 回の I/O 操作が必要だったことを表します。23 回の I/O 操作は #Bufs の表記における 24 回の I/O 操作に近いことに注意してください。
オプティマイザは、バックグラウンド・プロセスが正常に終了したのでフォアグラウンド・プロセスを中止しています（`ABA）。これで、オプティマイザは最終ステージのフェッチの準備ができました。

(7) ~E#0004.01(1) Fin Buf DBKeys=63 Fetches=0+0 RecsOut=63
この行は、最終（Fin）ステージではすでに 63 個の dbkey がすべてバッファ（Buf）に格納され、dbkey リストに対してテストされ、あらかじめ提供された dbkey リスト内の 63 個の dbkey がすべて無視されたことを表します。したがって、I/O 操作はありませ

ん（Fetches=0+0）。また、フォアグラウンド・プロセスをあわせると 63 行が提供されています（RecsOut=63）。

(8) 最終出力行は、常に対応する実行トレース行の後に印刷されます。

実行トレースを伴うソート順リーフ・ストラテジ

例 C-13（C-27）は、S フラグおよび E フラグの両方をセットした場合のソート順のリーフ検索ストラテジを説明しています。

例 C-13 SE フラグ・ディスプレイ --- ソート順検索

```
SQL> SELECT EMPLOYEE_ID, LAST_NAME
  1> FROM EMPLOYEES
  2> WHERE LAST_NAME = 'Clarke'
  3> ORDER BY EMPLOYEE_ID;
~S#0004 (1)
Leaf#01 Sorted EMPLOYEES Card=100 (2)
  FgrNdx  EMP_EMPLOYEE_ID [0:0] Fan=17 (3)
  BgrNdx1 EMP_LAST_NAME [1:1] Fan=12 (4)
~E#0004.01(1) BgrNdx1 EofData  DBKeys=3  Fetches=0+0  RecsOut=0 (5)
 EMPLOYEE_ID   LAST_NAME (6)
 00188         Clarke
 00196         Clarke
~E#0004.01(1) FgrNdx  Sorted   DBKeys=5  Fetches=0+4  RecsOut=3 (7)
 00212         Clarke (8)
3 rows selected
SQL>
```

次の番号は、例 C-13（C-27）の番号に対応しています。

(1) これは、セッションの 4 番目のクエリーです。

(2) これは、このクエリーの実行ツリー内にある最初（で唯一）のリーフ・ノードです。オプティマイザは、ソート順のリーフ・タイプを選択して、100 行のカーディナリティを持つ EMPLOYEES テーブルにアクセスします。

(3) フォアグラウンド・プロセスは、インデックス（FgrNdx）のスキャンに EMP_EMPLOYEE_ID を使用します。このインデックスは、指定したソート順で行を返します。クエリーは、EMPLOYEE_ID 列に対して条件を指定していないので、インデックス全体は、[0:0] 表記法で示されたようにスキャンされる必要があります。ファンアウト・ファクタは 17 です。

(4) バックグラウンド・プロセスは、1 つのインデックス（BgrNdx1）EMP_LAST_NAME を使用します。フォアグラウンド・プロセスが EMP_EMPLOYEE_ID をスキャンしている間に、バックグラウンド・プロセスは、同時に EMP_LAST_NAME インデックスをスキャンします。クエリーは、LAST_NAME 列に対して同じ値（= 'Clarke'）を指定しています。したがって、[1:1] 表示法は、1 つの下位 Ikey セグメントと 1 つの上位 Ikey

セグメントを示します。ファンアウト・ファクタは 12 です。
フォアグラウンド・プロセスは、EMP_EMPLOYEE_ID ソート・インデックス上で完全なスキャンを行います。スキャン中にバックグラウンドで dbkey リストがまったく利用できなければ、フォアグラウンド・プロセスは、dbkey に対してすべての行をフェッチします。ある時点で、完全なバックグラウンド dbkey リストが利用できるようであれば、フォアグラウンド・プロセスは、バックグラウンド dbkey リストに属している行のみをフェッチします。

(5) バックグラウンド・プロセスは、そのインデックスを最後（EofData）までスキャンして最初に終了します。バックグラウンド・プロセスには、オープンするためにインデックス・ページを取得する I/O 操作も、インデックス・スキャン用の I/O 操作も必要でない 3 つの dbkey があります。この時点では、フォアグラウンド・プロセスによって行はまったく渡されていません（RecsOut=0）。かわりに、dbkey が dbkey リストに書き込まれています。

(6) フォアグラウンド・プロセスは、行を渡します。

(7) フォアグラウンド・プロセスは、これでバックグラウンド・プロセスによって作成された dbkey リストを使用することができるので、テーブル内のすべての行をフェッチする必要はありません。フォアグラウンド・インデックスからの dbkey は、dbkey リストを通してフィルタ処理されます。dbkey があれば、フォアグラウンド・プロセスは行をフェッチして、完全なクエリー選択に対して一致させます。dbkey が dbkey リスト内になければ、行はフェッチされません。
フォアグラウンド・プロセスは、フォアグラウンド・インデックス（EMP_EMPLOYEE_ID [0:0]）全体をスキャンし、5 つの dbkey（DBKeys=5）を使用してデータ・レコードのフェッチを行います。バックグラウンド・プロセスの dbkey リストを通して EMP_EMPLOYEE_ID インデックス dbkey をフィルタし、4 ページの I/O 操作（Fetches=0+4）が必要な 3 行（RecsOut=3）を出力します。フォアグラウンド・インデックス・スキャンを行う 4 つのフェッチには、インデックス・スキャンそれ自身と必要なすべてのデータ・レコードのフェッチも含まれています（この場合、3 つの dbkey を使用）。バックグラウンド dbkey リストがしばらく利用できなければ、テーブル全体に対するすべての 100dbkey のフェッチが必要な場合があります。

(8) 最終的な出力行は、その引渡しに責任を持つため、常に実行トレース行の後で表示されます。

実行トレース付きのインデックスのみのリーフ・ストラテジ

例 C-14（C-29）は、S フラグおよび E フラグの両方をセットした場合のインデックスのみのリーフ検索ストラテジを説明しています。このリーフ・ストラテジを説明するため、この例では 2 つのインデックスを作成しています。

```
SQL> CREATE INDEX EMP_LN_FN_SX ON EMPLOYEES
  1> (LAST_NAME, FIRST_NAME, SEX);
SQL>
SQL> CREATE INDEX EMP_SX ON EMPLOYEES
```

```
  1> (SEX);
```

EMP_LN_FN_SX インデックスには、サンプルのクエリーで必要なすべての列が含まれています。

例 C-14 SE フラグの表示 --- インデックスのみのリーフ検索

```
SQL> SELECT LAST_NAME, FIRST_NAME FROM EMPLOYEES
  1> WHERE FIRST_NAME <> 'Alvin' AND LAST_NAME >= 'L'
  2> AND FIRST_NAME <= 'U' AND SEX = 'F';
~S#0004 (1)
Leaf#01 NdxOnly EMPLOYEES Card=100 (2)
  FgrNdx  EMP_LN_FN_SX [1:0] Fan=9 (3)
  BgrNdx1 EMP_SX [1:1] Fan=19 (4)
 LAST_NAME        FIRST_NAME (5)
 Lapointe         Hope
 Lapointe         Jo Ann
 MacDonald        Johanna
.
.
.
 Ulrich           Christine
 Villari          Christine
 Watters          Christine
~E#0004.01(1) FgrNdx  NdxOnly  DBKeys=16  Fetches=2+0  RecsOut=16 (6)
 Watters          Cora (7)
16 rows selected
SQL>
```

次の番号は、例 C-14（C-29）の番号に対応しています。

(1) これは、セッションの 4 番目のクエリーです。

(2) これは、このクエリーの実行ツリー内にある最初（で唯一）のリーフ・ノードです。オプティマイザは、インデックスのみ（NdxOnly）のリーフ・タイプを選択して、100 行のカーディナリティを持つ EMPLOYEES テーブルにアクセスします。

(3) フォアグラウンド・プロセスは、クエリーが必要なすべての列を含んでいるので、（この例のために特別に定義した）EMP_LN_FN_SX インデックスを選択します。1 つの low インデックス・キー値がありますが、スキャンに対する上限はありません。ファンアウト・ファクタは、各インデックス・ノードに対して平均 9 つのブランチがあることを示しています。

(4) バックグラウンド・プロセスは、EMPLOYEES テーブルの SEX 列をベースにした EMP_SX インデックス（この例のために特別に定義した）を選択します。クエリーは、SEX 列に対して同じ値（= 'F'）を指定しています。したがって、[1:1] 表示法は、1 つの下位 Ikey セグメントと 1 つの上位 Ikey セグメントを示します。ファンアウト・ファクタは 19 です。

(5) フォアグラウンド・プロセスは、すぐに列を返し始めます。

(6) フォアグラウンド・プロセスは、EMP_LN_FN_SX インデックスのスキャンを 2 度の物理 I/O 操作、16dbkey の検索および 16 行を返して完了します。フォアグラウンド・インデックスには、クエリーで必要なすべての行が含まれており、非常に効果的であるので、バックグラウンド・スキャンは開始されません。

(7) 最終的な出力行は、その引渡しに責任を持つため、常に実行トレース行の後で表示されます。

実行トレース付きソート順のリーフ・ストラテジを使用した結合

例 C-15（C-30）は、一致ストラテジのジグザグバリエーションを使用する 2 つのテーブルの結合を説明しています。1 つのループは、ソート順のリーフ検索ストラテジを使用し、2 番目のループは、インデックスのみの検索ストラテジを使用しています。S フラグと E フラグの両方が設定されています。

例 C-15 SE フラグの表示 --- 結合付きのリーフ・ストラテジ

```
SQL> SELECT E.EMPLOYEE_ID, JH.JOB_START
  1> FROM EMPLOYEES E, JOB_HISTORY JH
  2> WHERE E.EMPLOYEE_ID = JH.EMPLOYEE_ID
  3> AND (E.LAST_NAME > 'T' OR E.LAST_NAME > 'Y' OR E.LAST_NAME > 'Z');
~S#0005 (1)
Conjunct (10) (C-32)
Match (2)
  Outer loop (3)
    Leaf#01 Sorted EMPLOYEES Card=100 (4)
      FgrNdx  EMP_EMPLOYEE_ID [0:0] Fan=17 (5)
      BgrNdx1 EMP_LAST_NAME [1:0...]3 Fan=12 (6)
  Inner loop      (zig-zag) (7)
    Get     Retrieval by index of relation JOB_HISTORY (8)
      Index name  JH_EMPLOYEE_ID [0:0] (9) (C-32)
 E.EMPLOYEE_ID   JH.JOB_START
 00164            5-Jul-1980  (11) (C-32)
~E#0005.01(1) BgrNdx1 EofData  DBKeys=9  Fetches=1+0  RecsOut=1 (12) (C-32)
 00164           21-Sep-1981
 00170           26-Nov-1980  (13) (C-32)
 00186           25-Apr-1980
 00186            1-Jul-1975
 00186           16-Aug-1977
    .
    .
    .
 00242           12-May-1978
 00242           11-May-1980
 00247           19-Jan-1982
 00276            3-Jun-1977
```

```
~E#0005.01(1) FgrNdx  Sorted   DBKeys=11  Fetches=0+7  RecsOut=9 (14) (C-32)
 00276           15-Mar-1980 (15) (C-33)
20 rows selected
SQL>
```

次の番号は、例 C-15（C-30）の番号に対応しています。

(1) これは、セッションの 5 番目のクエリーです。

(2) このオプティマイザは、クエリー内の 2 つのテーブルの結合に一致ストラテジ（特にジグザグ・バリエーション）を使用します。一致ストラテジの 2 つのループは、別のテーブルにアクセスします。

(3) 外側のループに割り当てられているテーブルは、内側のループと連動してスキャンされます。2 つのループの処理は、各ループ内の現行のキーをできるだけ接近させながら進んでいきます。オプティマイザは低い値のキーを使用して、もう一方のループで低い値のキーに一致する（等しい）か、それを超える（より大きい）キーを検索します。

(4) オプティマイザは、ソート順のリーフ・タイプを選択して、100 行のカーディナリティを持つ EMPLOYEES テーブルにアクセスします。

(5) フォアグラウンド・プロセスは、EMP_EMPLOYEE_ID インデックス（FgrNdx）を使用します。このインデックスは、ソートした順序で行を返します。インデックス全体は、[0:0] 表記法で示されたようにスキャンされる必要があります。ファンアウト・ファクタは 17 です。

(6) バックグラウンド・プロセスは、1 つのインデックス（BgrNdx1）EMP_LAST_NAME を使用します。フォアグラウンド・プロセスが EMP_EMPLOYEE_ID をスキャンしている間に、バックグラウンド・プロセスは、同時に EMP_LAST_NAME インデックスをスキャンします。クエリーは、LAST_NAME 列に対して 3 つの範囲（> 'T' OR > 'Y' OR > 'Z'）を指定します。したがって、[1:0...]3 表記法は、3 つの範囲に対して、1 個の下位 Ikey セグメントと 0 個の上位 Ikey セグメントを示します。ファンアウト・ファクタは 12 です。
フォアグラウンド・プロセスは、ソート・インデックス EMP_EMPLOYEE_ID 上で完全なスキャンを行います。インデックス内の各 dbkey に対して、フォアグラウンドは、行をフェッチして、その行が EMPLOYEES テーブル上の 2 番目の条件である LAST_NAME > 'T' OR LAST_NAME > 'Y' OR LAST_NAME > 'Z' を満たすことを確認します。バックグラウンド・プロセスは、EMP_LAST_NAME インデックスを使用して、dbkey リストを構築することで支援します。dbkey リストは、LAST_NAME > 'T' OR LAST_NAME > 'Y' OR LAST_NAME > 'Z' 条件を満たす、すべての行の dbkey を含みます。バックグラウンド・インデックス・スキャンは、'T' より大きい値で起動するので、バックグラウンドが持つ dbkey はフォアグラウンドよりも少なく、フォアグラウンドが終了する前に終了します。フォアグラウンド・プロセスは、dbkey リストを通してその dbkey をフィルタ処理して、不必要な行のフェッチを防ぎます。

ソート順のリーフ・ストラテジを使用して、外部のループは内部のループに 1 つずつ E.EMPLOYEE_ID の値を渡します。

(7) 内部ループに割り当てたテーブルは、次にスキャンされます。ジグザグ・ストラテジは、内部ループに適用されます。

(8) オプティマイザは、内部ループに対する JOB_HISTORY テーブルへのインデックス・アクセスを選択します。

(9) オプティマイザは、インデックス JH_EMPLOYEE_ID を使用して、JOB_HISTORY テーブルへアクセスします。[0:0] 表記法は、インデックス全体をスキャンする必要があることを示します。外部ループから得られた E.EMPLOYEE_ID 値を使用して、オプティマイザは、JH_EMPLOYEE_ID インデックス内で一致するすべての値を検索します。

- 一致する値が見つからなければ、次に高い JH.EMPLOYEE_ID 値が外部ループに返され、そこでその値がソート順のリーフ・スキャンによって見つかった次の E.EMPLOYEE_ID 値と比較されます。
- 値が一致（JH_EMPLOYEE_ID = EMP_EMPLOYEE_ID）すれば、JOB_HISTORY テーブルの行がフェッチされ（前の行の Get で示された）、オプティマイザは重複した JH.EMPLOYEE_ID 値を得るために JH_EMPLOYEE_ID インデックスを継続してスキャンします。

EMPLOYEE_ID 値は、これ以上一致するものが見つからなくなるまでループの間を何度も（ジグザグ）通過します。

(10) Conjunct 表記法は、すべての一致ストラテジ上に表示されます。この例では、ジグザグ一致ストラテジが重複を避けるために渡す行をフィルタするので、一致ストラテジは必要ではありません。しかし、列のデータ型が異なれば、一致ストラテジは大まかな等価性チェックを実行し、Conjunct は、各ループからくる可能性がある一致キーの余分な対をフィルタ処理する必要があります。

(11) フォアグラウンド・プロセスは、バックグラウンド・プロセスがそのスキャンを完了する前にこの行を渡します。

(12) バックグラウンド・プロセスは、そのインデックス EMP_LAST_NAME を最後（EofData）までスキャンして、最初に終了します。バックグラウンド・プロセスは、インデックスのスキャンに対して 1 度の I/O 操作が必要な 9 つの dbkey を検索しました。9 つの dbkey を含む dbkey リストは、フォアグラウンド・プロセスがそのインデックスのスキャンを開始するときに、フォアグラウンド・プロセスに渡されます。フォアグラウンド・プロセスは、1 行（RecsOut=1）をフェッチして渡します。この点から、フォアグラウンド・プロセスは、行のフェッチに対して 9 つのバックグラウンド dbkey のみを考慮します。

(13) これらの行は、フォアグラウンド・プロセスによって渡されます。

(14) これによりフォアグラウンド・プロセスは、バックグラウンド・プロセスによって作成された dbkey リストを使用するので、テーブル内のすべての行をフェッチする必要はあ

りません。フォアグラウンド・インデックスからの dbkey は、dbkey リストを通してフィルタ処理されます。dbkey があれば、フォアグラウンド・プロセスは行をフェッチして、完全なクエリー選択に対して一致させます。dbkey が dbkey リスト内になければ、行はフェッチされません。
フォアグラウンド・プロセスは、バックグラウンド・プロセスの dbkey リストを通して EMP_EMPLOYEE_ID インデックス dbkey をフィルタし、7 ページの I/O 操作（Fetches=0+7）が必要な 9 行（RecsOut=9）を出力します。
9 行の出力行（RecsOut=9）は、実際の総出力の 20 行に一致していないことに注意してください。内部ループは、他の行の出力を持つ必要があり、これは重複した EMPLOYEE_ID 値（00164 および 00186）を示す結果から確認されます。

(15) 最終的な出力行は、その引渡しに責任を持つため、常に実行トレース行の後で表示されます。

C.7 R フラグによるソート統計の表示

OpenVMS VAX OpenVMS Alpha

OpenVMS では、Oracle Rdb 論理名 RDMS$DEBUG_FLAGS が R として定義された場合、または SORT_STATISTICS キーワード付きの SET FLAG 文がクエリーの実行中に有効である場合、Oracle Rdb はソート操作が完了した時点でソート統計を表示します。◆

Compaq Tru64 UNIX

SORT_STATISTICS キーワードおよび R デバッグ・フラグは、CompaqTru64 UNIX 上では無視されます。◆

OpenVMS VAX OpenVMS Alpha

SQL は、ORDER BY 句に対してソートを実行し、UNION および GROUP BY 句に必要な暗黙的なソートを行います。可能であれば、オプティマイザは適切なソート・インデックスが利用可能である場合やソート操作が冗長である場合、ソートを行いません。これらの場合、オプティマイザは、ソート操作を必要とせずに、行を正しい順番で検索する代替ストラテジを選択します。

オラクル社では、R デバッグ・フラグは常に S デバッグ・フラグとともに使用することをお薦めします（または、SET FLAGS 文の STRATEGY キーワード）。これを使用すると、ソート統計は制約、トリガー、システムまたはユーザー・クエリーなどの既知のクエリーと関連付けられます。さらに、SORT_STATISTICS キーワードは、STRATEGY 出力に余分な記述的情報を含ませることができます。

ソート操作は、CPU および I/O 操作の両方を消費します。デバッグ・フラグの出力を使用して、ソート作業ファイルの数と位置を調整でき、ソート仮想メモリーおよびいくつかのクエリーで必要なシステム・パラメータの分析を支援します。

例 C-16（C-33）は、mf_personnel データベース内の WORK_STATUS テーブルからのソート統計を示します。

例 C-16 サンプル・クエリーのストラテジ表示

```
SQL> SET FLAGS 'STRATEGY';
SQL> SELECT STATUS_CODE FROM WORK_STATUS ORDER BY STATUS_CODE;
```

```
Sort
Get     Retrieval sequentially of relation WORK_STATUS
 STATUS_CODE
 0
 1
 2
3 rows selected
```

SORT 統計は、1 つ以上の "Sort" キーワードがストラテジ表示内に表示された場合のみに生成されます。

このクエリーがインタラクティブ SQL またはダイナミック SQL で実行されたならば、SQL システム・クエリー（メタデータのロードと検証に使用される）も、ストラテジおよびソート統計を表示する可能性があります。したがって、SQL クエリーをユーザー・クエリーと区別することは重要です。

例 C-17（C-34）は、STRATEGY キーワードと SORT_STATISTICS キーワードの両方の使用を示します。

例 C-17 サンプル・クエリーに対するソート統計

```
SQL> SET FLAGS 'STRATEGY, SORT_STATISTICS';
SQL> SELECT STATUS_CODE FROM WORK_STATUS ORDER BY STATUS_CODE;
Sort
SortId# 4., # Keys 2 (1)
  Item# 1, Dtype: 2, Order: 0, Off: 0, Len: 1 (2)
  Item# 2, Dtype: 14, Order: 0, Off: 1, Len: 1 (3)
  LRL: 24, NoDups:0, Blks:5, EqlKey:0, WkFls: 2 (4)
Get     Retrieval sequentially of relation WORK_STATUS
 STATUS_CODE
 0
 1
SORT(2) SortId# 4, ---------------------- Version: V5-000 (5)
  Records Input: 3      Sorted: 3       Output: 0 (6)
LogRecLen Input: 24     Intern: 24      Output: 24 (7)
Nodes in SoTree: 112    Init Dispersion Runs: 0 (8)
Max Merge Order: 0      Numb.of Merge passes: 0
Work File Alloc: 0 (9)
MBC for Input: 0        MBC for Output: 0 (10)
MBF for Input: 0        MBF for Output: 0 (11)
Big Allocated Chunk: 88576 idle (12)
 2
3 rows selected
```

(1) STRATEGY キーワードが、SORT_STATISTICS とともに使用された場合、Oracle Rdb はソート操作についての追加情報を表示します。キーの数（# Keys）は、NULL インジケータとソートされる列の 2 つのデータ項目があることを示します。

(2) **Item# 1** は、UNSIGNED BYTE 型の NULL インジケータを説明します。これは、DESCENDING (1) 順ではなく、ASCENDING (0) 順です。これは、論理レコードのオフセット 0 で発生し、1 バイトの長さです。ソート・インタフェースを呼び出す場合に Oracle Rdb が使用するデータ型については、表 C-4（C-39）を参照してください。

(3) **Item# 2** は、CHARACTER 型で、ASCENDING (0) 順の列 STATUS_CODE を説明します。これは、論理レコードのオフセット 1 で発生し、1 バイトの長さです。ソート・インタフェースを呼び出す場合に Oracle Rdb が使用するデータ型については、表 C-4（C-39）を参照してください。

(4) この場合の **LRL**（論理レコード長）は、24 バイトです。Oracle Rdb は、NULL インジケータ用の領域、テーブルの列および 2 つの dbkey フィールドの空間を割り当てます。dbkey は、4 ワード境界に整列されます。データが他のソース（たとえば、UNION）から派生した場合、dbkey アクセスは不可能であり、これらの余分な dbkey フィールドは、論理レコードの一部ではありません。
NoDups: この値が 0 である場合、ソートは重複を許可します。値が 1 であれば、重複は許可されません。DISTINCT および UNION の両方が、この属性に 1 を設定してソート操作が重複行を破棄する方法を示す例 C-18（C-36）および例 C-20（C-38）を参照してください。
Blks: これは、ソートされるデータのサイズの見積りです。これは、クエリー（RDMS$DEBUG_FLAGS O または SET FLAGS ESTIMATES オプションによって表示される）に対するカーディナリティと論理レコード長（LRL）の見積りをベースにしています。
EqlKey: このフラグは、特別な等価コールバック・ルーチンがソート・インタフェースに渡されているかどうかを示します。このフラグが常に 0 であれば、特別なルーチンは何も使用されていないことを示します。
WkFls: この値は、OpenVMS 上の RDMS$BIND_SORT_WORKFILES 論理名で確立されたソート作業ファイルの数を反映しています。

(5) **SortId#** は、このソート操作がストラテジ表示内で使用されるソート操作に関連していることを示します。**Version** は、ソート・インタフェースの Oracle Rdb の内部バージョンです。

(6) **Records Input** は、ソート・インタフェースに渡される行の数を説明します。**Sorted** は、実際にソートされた行の数を説明します。ソート操作へのファイル・インタフェースを使用すると、入力中に行を省略できますが、この値は、レコード・インタフェースを使用するので、Oracle Rdb の **Records Input** 値と同じになります。Oracle Rdb に対する**出力**は、常に 0 になります。

(7) **LogRecLen Input** は、論理レコード長です。これは、列値と、クエリーによって使用される NULL インジケータをベースにして計算できます。ソート操作は、結合または計算の結果をソートする可能性があることに注意してください。Oracle Rdb は、レコード・ソート・アルゴリズムのみを使用するので、**Intern** および **Output** は、**LogRecLen Input** と常に同じです。

(8) **Nodes in SoTree**、**Init Dispersion Runs**、**Max Merge Order** および **Numb.of Merge passes** は、ソート操作を説明しています。

(9) **Work File Alloc** は、いくつの作業ファイルがソート操作で使用されたかを示します。0 値は、ソートがメモリー内で完了したことを示しています。

(10) **MBC for Input** および **MBC for Output** は、入力ファイル用の OpenVMS RMS マルチブロック・カウントです。Oracle Rdb は、ファイル・インタフェースではなく、ソート・レコード・インタフェースを使用するので、この値は常に 0（ゼロ）です。

(11) **MBF for Input** および **MBF for Output** は、入力ファイル用の OpenVMS RMS マルチバッファ・カウントです。Oracle Rdb は、ファイル・インタフェースではなく、ソート・レコード・インタフェースを使用するので、この値は常に 0（ゼロ）です。

(12) **Big Allocated Chunk** は、ソート機能に対して割り当てられている仮想メモリーの総計です。Oracle Rdb は、将来のソート要求に対してこのメモリーを保持し、小さすぎる場合は拡張します。割り当てたメモリーを保持することで、Oracle Rdb は、仮想メモリーの割当てや解放を行う OpenVMS メモリー・システム・サービスに対する超過呼出しを回避します。
"Idle" は、このメモリーのブロックがこのソート操作に必要ではないことを示し、"In Use" は、現在説明しているソート操作が、この割り当てた仮想メモリーの一部またはすべてを使用することを示します。
Oracle Rdb が **Big Allocated Chunk** を使用するかどうか、そして割り当てるバイト数は、クエリー・ソリューション（RDMS$DEBUG_FLAGS O または SET FLAGS ESTIMATES オプションによって得られる）の見積ったカーディナリティをベースにして判断します。ソートは、データベースから任意のレコードが検索される前にこの見積ったカーディナリティで初期化されます。これは、ソート操作が実際のソートの準備にそのソート・ツリーを設定したときに、多くのページ・フォルト数が発生する原因になります。ソリューション・カーディナリティは、検索される行の数をただ数学的に見積ったものです。これは、実際に返される行の数とは異なる可能性があります。

例 C-18（C-36）は、出力を個別の行のみに削減するために DISTINCT 句を使用した場合の出力の違いを示しています。

例 C-18 ソート属性上の DISTINCT 句の効果

```
SQL> SELECT DISTINCT STATUS_CODE
  1> FROM WORK_STATUS WHERE STATUS_CODE >= '1' ORDER BY 1;
Reduce  Sort
SortId# 8., # Keys 2
  Item# 1, Dtype: 2, Order: 0, Off: 0, Len: 1
  Item# 2, Dtype: 14, Order: 0, Off: 1, Len: 1
  LRL: 24, NoDups:1, Blks:5, EqlKey:0, WkFls: 2 (1)
Conjunct        Get     Retrieval sequentially of relation WORK_STATUS
 STATUS_CODE
 1
```

```
SORT(6) SortId# 8, ---------------------- Version: V5-000
  Records Input: 2      Sorted: 2       Output: 0
LogRecLen Input: 24     Intern: 24      Output: 24
Nodes in SoTree: 112    Init Dispersion Runs: 0
Max Merge Order: 0      Numb.of Merge passes: 0
Work File Alloc: 0
MBC for Input: 0        MBC for Output: 0
MBF for Input: 0        MBF for Output: 0
Big Allocated Chunk: 88576 idle
 2
2 rows selected
```

(1) **NoDups** が 1 に設定されていると、ソート・キーの重複値を持つ行は、破棄されることを示します。この変更は、クエリー内の DISTINCT 句に対して行われます（ストラテジ表示内の Reduce キーワードによって示されます）。

例 C-19（C-37）は、ソートがまったく必要でない場合の出力の変更を示します。オプティマイザは、見積りをベースにしてアクセス・ストラテジを生成するので、0（ゼロ）または 1 行が返されるような場合でも常にソート操作を生成します。そのような場合、ソート操作は避けることができ、わずかな CPU 時間を節約します。

例 C-19 ソート操作の回避

```
SQL> SELECT STATUS_CODE
  1> FROM WORK_STATUS
  2> WHERE STATUS_CODE >= '4' ORDER BY 1;
Sort
SortId# 5., # Keys 2
  Item# 1, Dtype: 2, Order: 0, Off: 0, Len: 1
  Item# 2, Dtype: 14, Order: 0, Off: 1, Len: 1
  LRL: 24, NoDups:0, Blks:5, EqlKey:0, WkFls: 2
Conjunct        Get     Retrieval sequentially of relation WORK_STATUS
SORT(3), SortId# 5 ---- Avoided (1)
 No records
0 rows selected
SQL> SELECT STATUS_CODE
  1> FROM WORK_STATUS
  2> WHERE STATUS_CODE >= '2' ORDER BY 1;
Sort
SortId# 6., # Keys 2
  Item# 1, Dtype: 2, Order: 0, Off: 0, Len: 1
  Item# 2, Dtype: 14, Order: 0, Off: 1, Len: 1
  LRL: 24, NoDups:0, Blks:5, EqlKey:0, WkFls: 2
Conjunct        Get     Retrieval sequentially of relation WORK_STATUS
SORT(4), SortId# 6 ---- Avoided (1)
 No records
 STATUS_CODE
 2
```

```
1 row selected
```

(1) 入力ストリームが 0 または 1 行を含む場合、ソート操作全体が回避される可能性があります。

例 C-20（C-38）は、ソート呼出しを生成する UNION の例を示します。

例 C-20 UNION を使用して SORT を生成するクエリー

```
SQL> SELECT LAST_NAME FROM EMPLOYEES
  1> UNION
  2> SELECT LAST_NAME FROM CANDIDATES
  3> LIMIT TO 10 ROWS;
Firstn  Reduce  Sort (1)
SortId# 3., # Keys 2
  Item# 1, Dtype: 2, Order: 0, Off: 0, Len: 1
  Item# 2, Dtype: 14, Order: 0, Off: 1, Len: 14
  LRL: 15, NoDups:1, Blks:6, EqlKey:0, WkFls: 2
Merge of 2 entries
  Merge block entry 1
  Get     Retrieval sequentially of relation EMPLOYEES
  Merge block entry 2
  Get     Retrieval sequentially of relation CANDIDATES
 LAST_NAME
 Ames
 Andriola
 Babbin
 Bartlett
 Belliveau
 Blount
 Boswick
 Boyd
 Brown
SORT(1) SortId# 3, ---------------------- Version: V5-000
  Records Input: 103    Sorted: 103     Output: 0 (2)
LogRecLen Input: 15     Intern: 15      Output: 15
Nodes in SoTree: 212    Init Dispersion Runs: 0
Max Merge Order: 0      Numb.of Merge passes: 0
Work File Alloc: 0
MBC for Input: 0        MBC for Output: 0
MBF for Input: 0        MBF for Output: 0
Big Allocated Chunk: 88576 idle
 Burton
10 rows selected (3)
```

(1) UNION は、一意な行のセットを返すように定義されています。したがって、この例では、マージされた行のセットがソートされ（**Sort** キーワード）、個別行のセットに削減され（**Reduce** キーワード）、最後にアプリケーションに渡されます。

UNION ALL 句は、2 つの選択文からすべての行を返し、通常はソート操作を含みません。

(2) これは、ソート操作に渡された正確なレコード数が 103 行であることを示します。

(3) インタラクティブな SQL からの出力は、10 行のみが返されることを示します。この例では、LIMIT TO 句は出力の量を制限するために使用されています。LIMIT TO（内の Firstn キーワードを参照）は、すべてのソートが完了した後で、常に最後に実行されます。

表 C-4（C-39）に、ソート・インタフェースで使用されるデータ型を示します。

表 C-4 ソート・インタフェースで使用されるデータ型

型名	値	SQL データ型
DSC$K_DTYPE_ADT	35	DATE（VMS および ANSI）、TIME、TIMESTAMP
DSC$K_DTYPE_B	6	TINYINT
DSC$K_DTYPE_BU	2	NULL インジケータで使用
DSC$K_DTYPE_F	10	REAL
DSC$K_DTYPE_G	27	DOUBLE PRECISION
DSC$K_DTYPE_L	8	INTEGER
DSC$K_DTYPE_Q	9	BIGINT、INTERVAL
DSC$K_DTYPE_T	14	CHARACTER
DSC$K_DTYPE_VT	37	CHARACTER VARYING (VARCHAR)
DSC$K_DTYPE_W	7	SMALLINT

◆

C.8 T フラグによるトランザクション・アクティビティの表示

RDMS$DEBUG_FLAGS 論理名または RDB_DEBUG_FLAGS 構成パラメータを T と定義した場合、Oracle Rdb は、SET（または DECLARE）TRANSACTION 文によって生成されたトランザクション・パラメータ・ブロックをダンプし、コミットまたはロール・バックしたときのトランザクションも表示します。この情報は、アプリケーションのトランザクション・アクティビティをトレースする場合に有効です。例 C-21（C-39）に、サンプル表示を示します。

例 C-21 トランザクション（T）フラグによるトランザクション・アクティビティの表示

```
SQL> SET TRANSACTION
  1>     READ WRITE
  2>     WAIT 30
```

```
  3>     RESERVING EMPLOYEES FOR PROTECTED READ,
  4>          DEPARTMENTS FOR EXCLUSIVE WRITE,
  5>          WORK_STATUS FOR SHARED READ
  6>     ISOLATION LEVEL SERIALIZABLE
  7>     EVALUATING EMPLOYEES_PRIMARY_EMPLOYEE_ID AT COMMIT TIME;
 Compile transaction on db: X00000001
~T Transaction Parameter Block: (len=6)
0000 (00000) TPB$K_VERSION = 1
0001 (00001) TPB$K_ISOLATION_LEVEL3 (serializable)
0002 (00002) TPB$K_WAIT_INTERVAL 30 seconds
0005 (00005) TPB$K_WRITE (read write)
 Start_transaction on db: X00000001
 Commit_transaction on db: X00000001
 Prepare_transaction on db: X00000001
~T Transaction Parameter Block: (len=77)
0000 (00000) TPB$K_VERSION = 1
0001 (00001) TPB$K_ISOLATION_LEVEL3 (serializable)
0002 (00002) TPB$K_WAIT_INTERVAL 30 seconds
0005 (00005) TPB$K_WRITE (read write)
0006 (00006) TPB$K_COMMIT_TIME (evaluating) "EMPLOYEES_PRIMARY_EMPLOYEE_ID"
0025 (00037) TPB$K_LOCK_READ (reserving) "EMPLOYEES" TPB$K_PROTECTED
0031 (00049) TPB$K_LOCK_WRITE (reserving) "DEPARTMENTS" TPB$K_EXCLUSIVE
003F (00063) TPB$K_LOCK_READ (reserving) "WORK_STATUS" TPB$K_SHARED
SQL> ROLLBACK;
 Rollback_transaction on db: X00000001
SQL>
```

トランザクションの起動、準備、コミットおよびロール・バックは、トランザクションを操作しているデータベース・ハンドルも示します。

表 C-5（C-40）は、RDMS$DEBUG_FLAGS 論理名または RDB_DEBUG_FLAGS 構成パラメータに "T" を設定したときに生成されるトランザクション・パラメータ・ブロック・コードと各コードに対する簡単な説明を示しています。

表 C-5 トランザクション・パラメータ・ブロック（TPB）情報

TPB コード	説明
TPB$K_WAIT	トランザクションは、無期限にロックを待っています。デッドロックが報告されています。
TPB$K_NOWAIT	トランザクションは、ロックを待っていませんが、ロックの競合が報告されています。
TPB$K_WAIT_INTERVAL	WAIT は、タイムアウト間隔とともに指定されているので、トランザクションは、指定した秒数待ちます。
TPB$K_READ	読取り専用トランザクションです。

表 C-5 トランザクション・パラメータ・ブロック（TPB）情報（続き）

TPB コード	説明
TPB$K_WRITE	読取り / 書込みトランザクションです。
TPB$K_BATCH_UPDATE	バッチ更新トランザクションです。
TPB$K_LOCK_READ	名前付きテーブルは、READ 用に予約されています。モードは、SHARED、PROTECTED または EXCLUSIVE できます。
TPB$K_LOCK_WRITE	名前付きテーブルは、WRITE 用に予約されています。モードは、SHARED、PROTECTED または EXCLUSIVE できます。
TPB$K_VERB_TIME	名前付き制約は、その評価時間を VERB TIME（NOT DEFERRABLE）に変更します。
TPB$K_COMMIT_TIME	名前付き制約は、その評価時間を COMMIT TIME（DEFERRABLE）に変更します。
TPB$K_ISOLATION_LEVEL1	分離レベル読込みは、コミットされます。
TPB$K_ISOLATION_LEVEL2	分離レベル・リピータブル・リード
TPB$K_ISOLATION_LEVEL3	分離レベル・シリアライズ可能
TPB$K_DEGREE3	CONSISTENCY LEVEL 3（シリアライズ可能）は、推奨できません。かわりに ISOLATION LEVEL SERIALIZABLE を使用してください。
TPB$K_DEGREE2	CONSISTENCY LEVEL 2（コミット読込み）は、推奨できません。かわりに ISOLATION LEVEL READ COMMITTED を使用してください。

RDMS$DEBUG_FLAGS または RDB_DEBUG_FLAGS を T と定義する場合、表示は、例 C-22（C-41）で示すように、データベースが無効なスナップショットを持っているので、読取り専用トランザクションが、読取り / 書込みトランザクションにアップグレードされた場合を示しています。

例 C-22 スナップショット・ファイルが無効である場合に、読取り / 書込みトランザクションにアップグレードされた読取り専用トランザクションの表示

```
SQL> ALTER DATABASE FILENAME mf_personnel SNAPSHOT DISABLED;
SQL> --
SQL> ATTACH 'FILENAME mf_personnel';
SQL> SET TRANSACTION READ ONLY;
 Compile transaction on db: X00000002
~T Transaction Parameter Block: (len=2)
```

```
0000 (00000) TPB$K_VERSION = 1
0001 (00001) TPB$K_READ (read only)
 Start_transaction on db: X00000002
~T Snapshots are disabled, READ ONLY converted to READ WRITE
SQL>
```

C.9 Xt フラグを使用した TRACE 制御文のログ

SQL TRACE 制御文は、RDMS$DEBUG_FLAGS 論理名または RDB_DEBUG_FLAGS 構成パラメータが Xt として定義された後で、ログ・ファイルに値を書き込みます。TRACE 制御文を使用すると、複数の値の式を指定できます。TRACE 制御文は、評価した各値式の各値をログ・ファイル内に保管します。SQL は、論理名 RDMS$DEBUG_FLAGS または構成パラメータ RDB_DEBUG_FLAGS が Xt になるように定義されている場合のみ、ログのトレースをオンにします。文字 X は、大文字である必要があり、文字 t は小文字である必要があります。TRACE 制御文は、RDMS$DEBUG_FLAGS または RDB_DEBUG_FLAGS が Xt として定義されていない場合は無効です。トレースのログは、複雑な複数文の手続きのデバッグを行う場合に有効です。

TRACE 制御文の詳細は、『Oracle RMU Reference Manual』を参照してください。

索引

A

B

I

L

M

N

O

P

Q

R

S

T

U

V

W

あ

い

え

お

か

き

<

け

こ

さ

し

す

せ

そ

た

ち

つ

て

と

ね

の

は

ひ

ふ

へ

ほ

ま

れ

ろ

わ