Oracle® Rdb

New and Changed Features for Oracle Rdb

Release 7.1 for OpenVMS Alpha

June 2001

Part No. A90804-01

Oracle Rdb New and Changed Features for Oracle Rdb, Release 7.1 for OpenVMS Alpha

Part No. A90804-01

Contents

3 SQL New and Changed Features

4 Using Row Caching on OpenVMS

6 Using the New RMU Show Statistics Screens

List of Examples

List of Figures

List of Tables

Send Us Your Comments

Oracle Rdb New and Changed Features for Oracle Rdb, Release 7.1 for OpenVMS Alpha

Part No. A90804-01

Oracle Corporation welcomes your comments and suggestions on the quality and usefulness of this document. Your input is an important part of the information used for revision.

- Did you find any errors?
- Is the information clearly presented?
- Do you need more information? If so, where?
- Are the examples correct? Do you need more examples?
- What features did you like most?

If you find any errors or have any other suggestions for improvement, please indicate the document title and part number, and the chapter, section, and page number (if available). You can send comments to us in the following ways:

- Electronic mail: nedc-doc_us@oracle.com
- FAX: 603-897-3825 Attn: Oracle Rdb
- Postal service:

  Oracle Corporation
  Oracle Rdb Documentation
  One Oracle Drive
  Nashua, NH 03062-2804
  USA

If you would like a reply, please give your name, address, telephone number, and (optionally) electronic mail address.

If you have problems with the software, please contact your local Oracle Support Services.

Preface

Oracle Rdb software is a general-purpose database management system based on the relational data model.

This manual describes new and changed features for this release of Oracle Rdb.

Intended Audience

This manual is intended for all users of Oracle Rdb and should be read to supplement information contained in the Oracle Rdb documentation set and release notes.

To benefit the most from this manual, you should be familiar with Oracle Rdb, data processing procedures, and basic database management concepts and terminology. Also, you must have a good working knowledge of the host operating system on which Oracle Rdb is running.

Structure

This manual contains six chapters:

Chapter 1	Summarizes all new features and technical changes.
Chapter 2	Summarizes the new and changed features that affect all Oracle Rdb interfaces.
Chapter 3	Describes in reference format the new and changed features of the SQL interface to Oracle Rdb.
Chapter 4	Describes in user's guide format the row cache feature.

Chapter 5	Describes in reference format the new and changed features of the RMU interface to Oracle Rdb.
Chapter 6	Describes in user's guide format the new RMU Show Statistics screen displays.

Related Documents

For more information, see the following manuals in the Oracle Rdb documentation set:

- *Oracle Rdb Installation and Configuration Guide* (Release 7.1)
- *Oracle Rdb Release Notes* (Release 7.1)
- *Oracle Rdb7 Oracle SQL Reference Manual*
- *Oracle Rdb7 Guide to SQL Programming*
- *Oracle Rdb7 Oracle RMU Reference Manual*
- *Oracle SQL/Services Release 7.1.5 Release Notes*
- *Oracle SQL/Services Release 7.1.5 Installation and Configuration Guide*

Conventions

In this manual, OpenVMS refers to the OpenVMS Alpha operating system.

Rdb refers to Oracle Rdb for OpenVMS.

Release 7.1 of Oracle Rdb software may be referred to as V7.1 or release 7.1.

Release 7.0 of Oracle Rdb software may be referred to as V7.0, Version 7.0, release 7.0, or Rdb7.

The SQL interface to Oracle Rdb is referred to as SQL. This interface is the Oracle Rdb implementation of the SQL standard adopted in 1999. This standard is referred to as the ANSI/ISO SQL standard or SQL:1999. See the *Oracle Rdb Release Notes* for more information.

In examples, an implied carriage return occurs at the end of each line, unless otherwise noted. You must press the Return key at the end of a line of input.

The following conventions are also used in this manual:

Convention	Meaning
. . .	Vertical ellipsis points in an example mean that information not directly related to the example has been omitted.
. . .	Horizontal ellipsis points in statements or commands mean that parts of the statement or command not directly related to the example have been omitted.
boldface text	Boldface type in text indicates a term defined in the text.
italic text	Italic type in text is used for emphasis.
< >	Angle brackets enclose user-supplied names.
[ ]	Brackets enclose optional clauses from which you can choose one or none.
$	The dollar sign represents the DIGITAL Command Language prompt in OpenVMS. This symbol indicates that the DCL interpreter is ready for input.

1 Summary of New and Changed Features

This chapter summarizes new and changed features in Oracle Rdb release 7.1. Additional features (and changes to features documented in this manual) may have been added after this manual was finalized. Such features are documented in the *Oracle Rdb Release Notes.*

1.1 New and Changed Features That Affect All Interfaces

This section summarizes new and changed features that affect all interfaces to Oracle Rdb. These features include:

- New logical names

  Oracle Rdb introduces several new logical names that do the following:

  – Control how Oracle Rdb attempts to initialize files with specific patterns

    See Section 2.2.1 for details.

  – Specify the size of the hash table for temporary tables

    See Section 2.2.2 for details.

  – Notify the operator in the event of a storage area extension

    See Section 2.2.3 for details.

  – Set debugging flags

    A new logical name has been added that allows you to specify the keywords from the SQL SET FLAGS statement as equivalence names. In addition, four new logical names have been added that specify the same behavior as the NOREVERSE_SCAN, NOTRANSITIVITY, NOZIGZAG_MATCH, and MAX_STABILITY keywords of the SET FLAGS statement. See Section 2.2.4 and the SET FLAGS Statement for details.

  - Direct .ruj file information to a global section

    See Section 2.2.5 for details.
  - Limit the number and size of AIJ initialization I/O buffers

    See Section 2.2.6 for details.
  - Disable multithreaded online storage area creation

    See Section 2.10 for details.
  - Identify an available secondary network link to a standby database through an alternate remote node name

    See Section 1.4.2 for details.
  - Set parameters for the row cache server (RCS)

    Several new logical names have been added to manage the RCS. See Section 4.12 for details.
  - Disable writing to the recovery-unit journal (RUJ) and after-image journal (AIJ) during certain operations

    See Section 2.2.8 for details.

- Row caching to enhance the performance of memory buffers

  This new feature allows all processes attached to a database to share a pool of record occurrences that reside in shared memory row caches. See Chapter 4 for details.

- Ability to specify where RUJ buffers will be allocated

  You can specify whether RUJ buffers will be allocated in global or local memory. The RUJ buffers used by each process are usually allocated in local virtual memory. With the introduction of row caching, these buffers now can be assigned to a shared global section (global memory), so that the recovery process can process this in-memory buffer and possibly avoid disk access. See the ALTER DATABASE Statement, the CREATE DATABASE Statement, and the IMPORT Statement for details.

- Ability to disable the after-image journal (.aij) file and the recovery-unit journal (.ruj) file during certain operations

  For details, see Section 2.2.8 and the entries for the LOGGING and NOLOGGING clauses in the following sections:

  - CREATE STORAGE MAP Statement

- – ALTER STORAGE MAP Statement
- – CREATE TABLE Statement
- – CREATE INDEX Statement
- – ALTER INDEX Statement

In addition, you can disable logging to the .ruj and .aij files for these operations by setting the logical name RDMS$CREATE_LAREA_NOLOGGING. See Section 2.2.8 for more information.

- Unique identifier in bugcheck dump files

  Oracle Rdb now includes a unique identifier in bugcheck dump files. This unique identifier is intended to allow bugcheck dumps to be more easily identified and possibly cataloged by users and by Oracle Corporation. See Section 2.3 for details.

- Optimizer zigzag strategy now uses a temporary relation

  Oracle Rdb might use a temporary relation for storing intermediate results fetched from the inner loop of a join. This temporary relation avoids a rescan of the index when performing duplicate processing for the outer loop and results in lower I/O cost for the zigzag match strategy. See Section 2.4 for details.

- Process image name written to monitor log on OpenVMS

  For local (nonremote) database attach requests, the process image name (the main image being run by the process) is now logged to the Oracle Rdb monitor log file. See Section 2.5 for details.

- Monitor process uses less ENQLM on OpenVMS

  The monitor process now uses less ENQLM when it holds locks. See Section 2.8 for details.

- The .ruj file size limit has been increased to about 4 billion disk blocks (the existing OpenVMS size limit)

- Permanent restriction exists on calling Oracle Rdb from a shared image on OpenVMS

  If code in the image initialization routine of a shared image makes any calls into Oracle Rdb through SQL or any other means, access violations or other unexpected behavior may occur if the Oracle Rdb images have not had a chance to do their own initialization. See Section 2.9 for details.

- Online creation of storage areas is now performed in parallel

Online storage area addition now initializes the pages of multiple storage areas in a multithreaded, or parallel, operation. Multithreaded storage area initialization permits multiple I/O operations to be issued to multiple devices. This usually reduces the amount of time needed to create and initialize the storage areas. See Section 2.10 for details.

- New system relations

  Oracle Rdb introduces the following new system relations:

  – RDB$SEQUENCE
  – RDB$PROFILES
  – RDB$GRANTED_PROFILES
  – RDB$TYPES
  – RDB$TYPE_FIELDS

  In the Oracle Rdb command-line help, see the ORACLE_RDB*nn* topic and select the System_Relations subtopic (where *nn* is the version number if using multiversion). Or, from the SQL interface, select the SQL*nn* topic and select the System_Relations subtopic (where *nn* is the version number if using multiversion).

- New information tables

  Information tables are special read-only tables that can be created in an Oracle Rdb release 7.1 database and used to retrieve database attributes that are not stored in the existing relational tables. Information tables allow interesting database information, which is currently stored in an internal format, to be displayed as a relational table. For additional information about these information tables, see Section 2.1.

1.2 SQL New and Changed Features

This section summarizes new and changed features for the SQL interface to Oracle Rdb. The new and changed features include:

- Support for several new character set names as synonyms for Oracle Rdb character sets

  See Section 3.2.1.2 for details.

- Support for the ARABIC character set

Oracle Rdb supports the use of the ARABIC character set, as defined by the ASMO 449 and ISO9036 standards. It cannot be used as an identifier character set because the ARABIC character set does not include the ASCII character range.

```
ARABIC
1 octet
underscore = %X'5F'
percent = %X'25'
```

- Support for the UNSPECIFIED character set

  Oracle Rdb supports the use of the UNSPECIFIED character set. You can make comparisons and assignments between text objects (columns, literals, and so on) that have the UNSPECIFIED character set and any other text object regardless of the character set of the other text object. See Section 3.2.1.1 for details.

- Support for Oracle server data type keywords

  This release of Oracle Rdb adds support for several Oracle server data type keywords as synonyms for Oracle Rdb SQL keywords. See Section 3.2.2.1 for details.

- Support for the NUMBER data type

  Oracle Rdb now supports the NUMBER data type. See Section 3.2.2.2 for details.

- Support for LIST OF BYTE VARYING subtypes

  Oracle Rdb supports BINARY and TEXT as subtypes for the LIST OF BYTE VARYING data type. See Section 3.2.2.4 for details.

- Value expressions can be used in GROUP BY and ORDER BY clauses

  Value expressions are now supported in the select expression GROUP BY and ORDER BY clauses. See Section 3.2.5 for details.

- New data types are supported by the SQL precompiler

  The SQL precompiler now supports the following data types:

  – On OpenVMS Alpha, INTEGER*8 and LOGICAL*8 FORTRAN types, and __int64 and int64 C types

  – On OpenVMS Alpha, INTEGER*1 (8-bit binary) FORTRAN type, int8, int16, __int16, int32, and __int32 C types

  See Section 3.5.1 for details.

- UNION ALL and derived tables allow up to 2000 value expressions in select list

  Prior to Oracle Rdb release 7.0.2, the UNION ALL clause of the SELECT statement and derived tables were restricted to 255 value expressions. Beginning with Oracle Rdb release 7.0.2, the UNION ALL clause allows up to 2000 value expressions, and up to 2000 columns are visible through a derived table expression when you access an Oracle Rdb release 7.0.2 or later database.

- SET DIALECT statement affects server behavior after an ATTACH statement

  The SET DIALECT statement now affects server behavior after you issue an ATTACH statement. In previous releases of Oracle Rdb, you were required to set the dialect before executing an ATTACH statement to get SQL92, SQL99, or Oracle Level1 behavior for some statements.

- When an Oracle Rdb process writes a bugcheck dump file, the system operator classes CLUSTER and CENTRAL are now notified. See Section 2.2.7 for details.

- Create, alter, or drop special user entries

  The new CREATE USER, ALTER USER, and DROP USER statements allow you to create, alter, or drop a special entry that identifies a database user. That user can be granted roles that provide access to database objects. See the following sections for details on creating, altering, and dropping users:

  – CREATE USER Statement

  – ALTER USER Statement

  – DROP USER Statement

- Automatically generate primary key values

  The new CREATE SEQUENCE statement provides a means by which to automatically generate primary key values. The new ALTER SEQUENCE and DROP SEQUENCE statements allow you to alter or drop a sequence created with the CREATE SEQUENCE statement. See the following sections for details:

  – CREATE SEQUENCE Statement

  – ALTER SEQUENCE Statement

  – DROP SEQUENCE Statement

  You create slots for sequences in the database root file with the new RESERVE n SEQUENCES clause on the CREATE DATABASE or ALTER DATABASE statement. See the following sections for details:

  – CREATE DATABASE Statement

  – ALTER DATABASE Statement

- Create, alter, or drop profiles

  Profiles are used to extend the user definition within the database with special attributes that control transactions and resource usage. When a user connects to the database, user information is loaded that may include a specified profile.

  Consider the example where user SMITH is a decision-support user who is permitted to query production data when making decisions for a company. The database administrator would like to prevent SMITH from accidentally starting a read/write transaction (because this affects the locking and I/O to the snapshot files). A special profile is created for user SMITH and other users in his department. For example:

```
SQL> CREATE PROFILE DECISION_SUPPORT
cont>   COMMENT IS 'limit transactions used by report writers'
cont>   TRANSACTION MODES (NO READ WRITE, READ ONLY);

SQL> CREATE USER SMITH PROFILE DECISION_SUPPORT;
```

  When user SMITH connects and attempts to start a READ WRITE transaction, he receives an error as shown in the following example:

```
SQL> SET TRANSACTION READ WRITE;
%RDB-E-BAD_TPB_CONTENT, invalid transaction parameters in the transaction
parameter block (TPB)
-RDMS-E-INVTRANOPT, the transaction option "READ WRITE" is not allowed
```

  Further, the database administrator can provide a default transaction to make it easier for SMITH to perform queries, for example:

```
SQL> ALTER PROFILE DECISION_SUPPORT
cont>   DEFAULT TRANSACTION READ ONLY;
```

  When SMITH starts a session, he can use the explicit START DEFAULT TRANSACTION syntax, or have SQL automatically start a default transaction which is READ ONLY. See the following sections for details:

  – CREATE PROFILE Statement

  – ALTER PROFILE Statement

  – DROP PROFILE Statement

- Create, alter, or drop synonyms

Synonyms are special objects that provide an alternate name for an existing database object. For instance, a stored procedure named ADD_NEW_EMPLOYEE might have an alternate name NEW_EMP. There are many reasons why a database administrator might want to use a synonym:

- Shorter names make it easier for interactive SQL users to type the names of objects.
- When tables or routines change names, synonyms can be used to allow existing applications to run unchanged.
- Changes in corporate naming schemes might require table, domain, and routine name changes. This can be done gradually by defining synonyms for these objects that conform to the new scheme, and so allow new applications to be written, while older applications are changed to use the new name. Eventually the synonyms are removed.
- If applications use the synonym, it is possible to substitute tables or routines over time. Initially the synonym TRANSACTION references TRANS_001, at the end of the month the synonym is altered to reference an identically defined table called TRANS_002, and so on.

- Create, alter, or drop roles

  The new CREATE ROLE, ALTER ROLE, and DROP ROLE statements allow you to create, alter, or drop a database object to which privileges and other roles can be granted. A role can be granted to a user or another role. By creating roles and granting privileges to that role, you can, for example, create a role for members of a department. When a user leaves the department, the departmental role can be revoked from that user, thus excluding that user's access to the departmental files. See the following sections for details:

  - CREATE ROLE Statement
  - ALTER ROLE Statement
  - DROP ROLE Statement

- Grant and revoke roles

  A new form of the GRANT and REVOKE statements allows you to grant a role to a user, to another role, or to the PUBLIC user. See the GRANT Statement: Roles and the REVOKE Statement: Roles sections for details.

- Wildcard character is valid in GRANT and REVOKE statements

The GRANT and REVOKE statements now support the use of the wildcard character to specify all objects of a specific type. See the following sections for details:

- GRANT Statement
- GRANT Statement: ANSI/ISO-Style
- REVOKE Statement
- REVOKE Statement: ANSI/ISO-Style

- Create or add privileges to an entry of the Oracle Rdb access control list for a sequence

  New clauses on the GRANT and GRANT: ANSI/ISO-Style statements allow you to create or add privileges to an entry of the Oracle Rdb access control list for a sequence. See the following sections for details:

  - GRANT Statement
  - GRANT Statement: ANSI/ISO-Style

- Remove privileges from an entry in the Oracle Rdb access privilege set for a sequence

  New clauses on the REVOKE and REVOKE: ANSI/ISO-Style statements allow you to remove privileges from an entry of the Oracle Rdb access control list for a sequence. See the following sections for details:

  - REVOKE Statement
  - REVOKE Statement: ANSI/ISO-Style

- SHOW statement supports new database objects

  With this release of Oracle Rdb, you can create users, roles, and sequences. The SHOW statement has been updated to display these new objects. See the SHOW Statement for details.

- Load LIST OF BYTE VARYING data from a file without special application code

  The new INSERT from FILENAME statement allows you to load a column of the LIST OF BYTE VARYING data type from a text or binary file without using special application code. The specified file is opened and each record is read and stored in the LIST OF BYTE VARYING column specified by the list cursor. See the INSERT from FILENAME Statement for details.

- Lock tables when the table is not known when the transaction begins

The new LOCK TABLE statement allows tables to be locked at modes other than SHARED READ when the table is not known when the transaction begins. See the LOCK TABLE Statement for details.

- Reuse the current database attach for another user

  The new SET SESSION AUTHORIZATION statement allows you to transfer the current database attach to another user. See the SET SESSION AUTHORIZATION Statement for details.

- New GREATEST and LEAST value expressions

  These value expressions return, respectively, the greatest or least value from a list of value expressions. These functions are provided for compatibility with Oracle servers. See Section 3.2.3.1 for details.

- Unique predicate support

  The new UNIQUE predicate allows you to determine if duplicate rows exist in the result table of a column select expression. See Section 3.2.4.1 for details.

- New compound statements

  The following compound statements have been added to this release:

  – CASE (Searched) Control Statement
  – FOR (Counted) Control Statement
  – ITERATE Control Statement
  – REPEAT Control Statement
  – WHILE Control Statement

- Enhancements to existing compound statements

  Several existing compound statements have been enhanced, as follows:

  – The CASE (simple) control statement now allows a list of value expressions.
  – The LEAVE control statement allows the label to be omitted.
  – The SIGNAL control statement allows SQL99 syntax and an optional character parameter for additional diagnostics.
  – Compound statements now support the OPTIMIZE clause.

  See the following sections for details:

  – CASE (Searched) Control Statement

- LEAVE Control Statement
- SIGNAL Control Statement
- Compound Statement

- Ability to add comments when database objects are created or altered

  The syntax for the following statements has been extended to allow you to add a comment with the COMMENT IS clause when the object of the operation is created or altered:

  - CREATE STORAGE MAP Statement and ALTER STORAGE MAP Statement

    The COMMENT IS clause can also be specified for each storage area in an index-store-clause
  - CREATE INDEX Statement and ALTER INDEX Statement

    The COMMENT IS clause can also be specified for each storage area in an index-store-clause
  - CREATE TABLE Statement and ALTER TABLE Statement
  - ALTER CONSTRAINT

    See *Oracle Rdb Release Notes*.
  - ALTER TRIGGER Statement

    See *Oracle Rdb Release Notes*.

- INSERT statement allows you to insert default values

  You can now specify the DEFAULT VALUES clause to indicate that you want column default values inserted into a table. See the INSERT Statement for details.

- Specify thresholds for LIST OF BYTE VARYING storage maps and as an index attribute

  Support has been added that allows you to specify thresholds for both LIST OF BYTE VARYING storage maps and as an attribute of an index. See the following sections for details:

  - CREATE STORAGE MAP Statement
  - ALTER STORAGE MAP Statement
  - CREATE INDEX Statement

  - ALTER INDEX Statement

- Specify names for vertical and horizontal partitions

  The syntax for CREATE STORAGE MAP has been enhanced to allow you to name vertical partitions and the CREATE STORAGE MAP, ALTER STORAGE MAP, CREATE INDEX, and ALTER INDEX statements have been enhanced to allow you to name horizontal partitions. Oracle Rdb uses partition names for operations such as ALTER INDEX ... DROP PARTITION. See the following sections for details:

  - CREATE STORAGE MAP Statement
  - ALTER STORAGE MAP Statement
  - CREATE INDEX Statement
  - ALTER INDEX Statement

- ALTER INDEX statement lets you add, drop, move, and rename partitions

  The ALTER INDEX statement provides several new clauses for partition management. See the ALTER INDEX Statement for details.

- Enable or disable triggers or constraints when creating or altering a table or trigger

  New syntax has been added to allow you to enable or disable triggers, constraints, or both when you create or alter a table or trigger. In addition, when you enable constraints as part of a CREATE TABLE or ALTER TABLE statement, you can disable constraint revalidation. See the following sections for details:

  - CREATE TABLE Statement
  - ALTER TABLE Statement
  - ALTER TRIGGER Statement

- Reorder the display of table column positions

  New syntax has been added to the ALTER TABLE statement that allows you to specify the order in which table columns should be displayed when a SHOW TABLE statement or select * expression is issued. However, note that the on-disk layout is not changed when you use this new syntax. See the ALTER TABLE Statement for details.

- Specify automatic columns for insert and update operations

Automatic columns are a new type of read-only column that can be used to provide a complex default for a column when a row is inserted, updated, or both. See the CREATE TABLE Statement and the ALTER TABLE Statement for details.

- COMMENT ON statement extended to apply to additional database objects

  The COMMENT ON statement syntax has been enhanced to allow you to apply comments to collating sequences, constraints, functions, modules, procedures, roles, sequences, storage maps, triggers, users, views, databases, and all columns of a table. See the COMMENT ON Statement for details.

- Support for SQL92 or later constraint attribute syntax

  Prior to this release, the constraint attribute syntax for the ALTER TABLE statements did not conform to the SQL92 or later standard. This syntax has been enhanced for this release of Oracle Rdb to conform to the standard. See the CREATE TABLE Statement and the ALTER TABLE Statement for details.

- Support for a UNIQUE constraint that complies with intermediate-level SQL92 or later

  Oracle Rdb now provides an UNIQUE constraint that complies with intermediate-level SQL92 or later. This type of constraint excludes NULL columns from the UNIQUE comparison. See the CREATE TABLE Statement for details.

- External routine enhancements

  Several enhancements have been made to external routines and functions, as follows:

  – Syntax has been updated to reflect the SQL / PSM standard for PARAMETER STYLE GENERAL.

  – New RETURNS NULL ON NULL INPUT and CALLED ON NULL INPUT clauses have been added that allow you to control how an external function is invoked when one or more of the arguments are null.

  – DETERMINISTIC and NOT DETERMINISTIC clauses have been added as synonyms for the NOT VARIANT and VARIANT clauses for conformance to the SQL / PSM standard.

  – Parameters defined for a function or procedure with mode IN can be given default values.

  – Parameters for a function or procedure can include a COMMENT IS clause, so that each parameter can be documented.

See the Create Routine Statement and the CREATE MODULE Statement for details.

- Reserve or lock horizontal partitions

  The SET TRANSACTION statement provides a new PARITITION option in the RESERVING clause that allows you to reserve or lock one or more partitions of a horizontally partitioned table. The purpose is to provide concurrent partitioned operations on a single table with the highest locking modes available. See the SET TRANSACTION Statement for details.

- Enhancements to DROP STORAGE AREA CASCADE clause

  The DROP STORAGE AREA CASCADE clause of the ALTER DATABASE statement has been enhanced as follows:

  – You can now drop a storage area that contains a whole index in a single area.

  – Constraints are not validated if it is not warranted.

  – Debugging output for DROP STORAGE AREA CASCADE is now available.

  See the ALTER DATABASE Statement for details.

- The ALTER INDEX statement performs fewer area scans

  When the ALTER INDEX statement is used to change the partitioning of an index, it scans each referenced storage area to initialize and create the index partitions.

  The amount of I/O has been reduced (especially for mixed-format areas) by avoiding one of the area scans. Prior to this change, Oracle Rdb would scan each of the areas twice to remove the old index. Now, Oracle Rdb performs only a single scan of each area to remove the old hash index.

  This change uses the same optimizations introduced in Rdb7 for statements such as TRUNCATE TABLE, DROP TABLE, DROP INDEX, and ALTER DATABASE ... DROP STORAGE AREA CASCADE.

- CONSTRAINT clause for ALTER TABLE now conforms to the SQL92 or later standard

  In Oracle Rdb, the syntax for adding a constraint definition to a table includes the clause ADD CONSTRAINT CONSTRAINT <name>. In the SQL92 or later standard, the syntax includes the clause ADD CONSTRAINT <name>.

Specifically, in the SQL92 or later standard, CONSTRAINT is specified only once.

For compatibility, SQL now allows both ADD CONSTRAINT CONSTRAINT and ADD CONSTRAINT as valid syntax for the ALTER TABLE statement. See the ALTER TABLE Statement for details.

- New annotations on query outlines

  The order of the queries in an outline now matches the order of optimization, not the order of execution. Therefore, to make reading query outlines easier, comments are embedded in the outline to indicate the type of statement that generated the query component. See the CREATE OUTLINE Statement for details.

- Queries in the TRACE statement can be merged with the query outline

  If the TRACE statement is activated by the RDMS$DEBUG_FLAGS "Xt" (or by the SET FLAGS statement), then queries in the TRACE statement are merged into the query outline for the procedure. See the TRACE Control Statement for details.

- New keyword for the SET FLAGS statement improves some optimizer estimates

  The new INDEX_COLUMN_GROUP keyword activates the optimizer to make better estimates when the selection predicate in a query specifies only some of the leading segments in a multisegment index. See the SET FLAGS Statement for details.

- New keyword for the SET FLAGS statement for setting query outline mode

  The new MODE keyword allows you to set the query outline mode from within interactive and dynamic SQL. See the SET FLAGS Statement for details.

- Additional triggered statements supported for CREATE TRIGGER

  You can now specify the CALL statement, the SIGNAL control statement, and the TRACE statement as triggered statements in the CREATE TRIGGER statement. See the CREATE TRIGGER Statement for details.

- DECLARE Variable statement supported for CREATE MODULE

  You can specify the DECLARE Variable statement as part of the CREATE MODULE statement to specify a global variable. You can use such a variable to exchange information between routines. See the CREATE MODULE Statement for details.

- Control error reporting when a COMMIT or ROLLBACK statement is executed in the absence of an active transaction

  You can specify whether or not SQL should return an error when a COMMIT or ROLLBACK statement is executed in the absence of an active transaction. See Section 3.3.1, Section 3.5, and the SET QUIET COMMIT Statement for details.

- Enable or disable starting a transaction for procedures

  You can control the SQL behavior for starting default transactions for compound statements. See Section 3.3.1 and the SET COMPOUND TRANSACTIONS Statement for details.

- LANGUAGE SQL clause optional in CREATE MODULE statement

  The LANGUAGE SQL clause is now optional for module, procedure, and function definitions. See the CREATE MODULE Statement for details.

- The FROM keyword in the DELETE statement is optional

  The FROM keyword is now optional for the DELETE statement. See the DELETE Statement for details.

- Default node size displayed after index is created

  In prior releases of Oracle Rdb, a CREATE INDEX statement supplied a default index node size if none was provided for a unique sorted index or a sorted ranked index. However, neither the SQL SHOW INDEX or SHOW TABLE statement nor the RMU Extract command displayed the value of this default node size. Beginning with this release, all new indexes store the default node size for display by SQL and RMU. See the CREATE INDEX Statement for details.

- Edit strings can be associated with a value expression in SELECT statements

  A new EDIT USING clause is added to the SELECT statement so that you can associate an edit string with a value expression or domain name. See Section 3.2.5 for details.

- Date subtraction supported

  Some Oracle applications rely on being able to subtract one date from another to have the number of days between the two dates returned. To better support those applications, Oracle Rdb provides that support in the Oracle Level1 dialect. However, note that unlike Oracle database servers, partial days are not returned. The value returned is always an integer.

  The following example shows the subtraction of dates:

```
SQL> SET DIALECT 'ORACLE LEVEL1';
SQL> SELECT SYSDATE - DATE VMS '12-JAN-1998' FROM RDB$DATABASE;
          15
1 row selected
SQL>
```

- DECLARE variable statement propagates edits strings

  The DECLARE variable statement now propagates the EDIT STRING clause from a domain to a variable. You can display results assigned to the variable in the format defined by the edit string with the PRINT, SELECT, and CALL statements.

  The following example shows that the EDIT STRING is inherited from the domain used to defined the data type for the variable.

```
SQL> CREATE DOMAIN MONEY INTEGER(2)
cont> EDIT STRING '$$$,$$$,$$9.99';
SQL> DECLARE :AVG_SALARY MONEY;
SQL> --Calculate the average salary for all current jobs
SQL> SELECT AVG(SALARY_AMOUNT)
cont> INTO :AVG_SALARY
cont> FROM SALARY_HISTORY
cont> WHERE SALARY_END IS NULL;
SQL> PRINT 'The average salary is ', :avg_salary;
                              AVG_SALARY
 The average salary is        $31,922.79
```

- Edit strings applied to columns in SELECT statements that contain UNION and UNION ALL clauses

  Edit strings are now applied to columns that appear in a SELECT statement that contains UNION and UNION ALL clauses. In previous versions of Oracle Rdb, such EDIT STRING clauses were ignored. Note that only the edit strings in the first branch of the UNION are used. If none are present for a numeric value, then a default value is assigned. For example:

```
SQL> SELECT SALARY_AMOUNT, SALARY_START
cont> FROM SALARY_HISTORY
cont> WHERE EMPLOYEE_ID='00164'
cont> UNION
cont> SELECT 100.00, JOB_START
cont> FROM JOB_HISTORY
cont> WHERE EMPLOYEE_ID='00164';
 SALARY_AMOUNT   SALARY_START
       $100.00    5-Jul-1980
```

```
       $100.00   21-Sep-1981
    $26,291.00    5-Jul-1980
    $26,291.00    2-Mar-1981
    $50,000.00   21-Sep-1981
    $51,712.00   14-Jan-1983
6 rows selected
```

- Behavior of RESTRICTED ACCESS clause changed for the ATTACH statement

  By default, a transaction that reserves a table for EXCLUSIVE access does not reserve the LIST (segmented string) area for exclusive access. Because the LIST area is usually shared by many tables, SHARED access is assumed by default to permit updates to the other tables.

  This means that when you run an Oracle RMU load operation or an application updates a table reserved for EXCLUSIVE access, you might notice that the snapshot storage area (.snp) grows. This is because of the I/O to the LIST area that is performed by default when SHARED WRITE mode is in use.

  However, if you attach to the database using an SQL ATTACH statement and you specify the RESTRICTED ACCESS clause, then all storage areas are accessed in EXCLUSIVE mode. Use this clause to eliminate the snapshot I/O and related overhead if you are performing a lot of I/O to the LIST storage areas (for example, when you are restructuring the database, or dropping a large table containing LIST OF BYTE VARYING columns and data).

  Because RESTRICTED ACCESS is the default for the SQL IMPORT statement, there is reduced overhead during an import of LIST OF BYTE VARYING data.

- New SQLSTATE value

  If an SQL statement expects a value from a function that does not return a value, then the SQLSTATE value is set to '2F001' to reflect the error state.

  This new error code is shown in the following example.

```
SQL> CREATE DATABASE FILE TEST2;
SQL>      SET DIALECT 'SQL92';
SQL>
SQL>      CREATE MODULE RETURN_M
cont>           LANGUAGE sql
cont>
cont>           FUNCTION RETURN_F (:A INTEGER)
cont>            RETURNS INTEGER;
cont>           BEGIN
cont>           IF :A IS NOT NULL THEN
cont>         RETURN - :A;
```

```
cont>           END IF;
cont>           END;
cont>       END MODULE;
SQL>
SQL>      SELECT RETURN_F (NULL) FROM RDB$DATABASE;
%RDB-F-NORESULT, stored function "RETURN_F" returned no result
-RDB-F-ON_DB, on database SQL_USER4:[USER.DB]TEST2.RDB;
SQL> SHOW SQLCA
SQLCA:
        SQLCAID:        SQLCA           SQLCABC:        128
        SQLCODE:        -1043
        SQLERRD:        [0]: 0
                        [1]: 0
                        [2]: 0
                        [3]: 0
                        [4]: 0
                        [5]: 0
        SQLWARN0:       0       SQLWARN1:       0       SQLWARN2:       0
        SQLWARN3:       0       SQLWARN4:       0       SQLWARN5:       0
        SQLWARN6:       0       SQLWARN7:       0
        SQLSTATE:       2F001
SQL> ROLLBACK;
SQL> DROP DATABASE FILE TEST2;
```

See the *Oracle Rdb7 SQL Reference Manual* for a description of SQLSTATE and its other values.

1.3 RMU New and Changed Features

Oracle Rdb provides the Oracle RMU interface to manage, monitor, and display information about Oracle Rdb databases. This section summarizes new and changed features for the Oracle RMU interface to Oracle Rdb. New Oracle RMU features include:

- Change in behavior for the RMU Backup command Exclude and Include qualifiers

  The RMU Backup command no longer accepts the Include and Exclude qualifiers in the same command. In addition, the Exclude=* qualifier results in a root-file-only backup operation. Prior to this release, the Exclude=* qualifier caused all storage areas to be backed up. These changes were made to clarify what gets backed up when these qualifiers are used. See the RMU Backup Command for details.

- Backup operation to multiple disk files

  The RMU Backup command now allows you to specify that multiple disk files can be used for a backup operation. See the RMU Backup Command for details.

- Specify where prompts are displayed

  Several RMU commands now let you specify where server prompts should be displayed (client, server, or standard input devices).

  See the following sections for details:

  – RMU Backup Command
  – RMU Backup After_Journal Command
  – RMU Dump After_Journal Command
  – RMU Dump Backup_File Command
  – RMU Recover Command
  – RMU Restore Command

- Extract profile, role, sequence, synonym, and user definitions

  The RMU Extract command allows you to display profile, role, sequence, synonym, or user definitions as specified with the following commands:

  – SQL CREATE PROFILE or ALTER PROFILE
  – SQL CREATE ROLE or ALTER ROLE
  – SQL CREATE SEQUENCE or ALTER SEQUENCE
  – SQL CREATE SYNONYM or ALTER SYNONYM
  – SQL CREATE USER or ALTER USER

  See the RMU Extract Command for details.

- Default behavior of Checksum_Verification qualifier for the RMU Backup command changed

  In previous releases, the default value for the Checksum_Verification qualifier was Nochecksum_Verification. Beginning with this release, the default value is Checksum_Verification. The Checksum_Verification qualifier requests that the Oracle RMU command verify the checksum stored on each database page before the Oracle RMU backup operation is applied, thereby providing end-to-end error detection on the database I/O. See the RMU Backup Command for details.

- Extract items in alphabetic order or in approximate order of creation

  You can now specify how you want the RMU Extract command to display the storage area, cache, and journal names for the Database, Alter_Database (also known as Change_Database), and Import items. You can specify either alphabetic order (by the ASCII collating sequence) or approximate definition order. See the RMU Extract Command for details.

- Load a database under restricted access

  The RMU Load command provides a new qualifier, Restricted_Access, that allows a single process to load data and enables some optimizations available only when restricted access is in use. See the RMU Load Command for details.

- Update the root file with AIP logical area types

  The RMU Show Statistics utility needs to know the type of a logical area to display information on a per-logical-area basis. However, prior to Oracle Rdb release 7.0A, this data was not stored in the database root file. You can use a new option in the Initialize=Larea_Parameters=options-file qualifier of the RMU Repair command to store this information in the root file. If you do not make these updates, the Show Statistics utility will request a logical area type each time you request per-logical-area information for a logical area. See the RMU Repair Command for details.

- Display all nodes in the cluster where the database is open

  The RMU Show Users command has been enhanced to identify the various nodes in a VMScluster where the database is currently open and available for use. Note that the term open includes nodes where the database is open for utility access only.

- Enhanced header information in RMU Analyze output

  The RMU Analyze command now prints the current date and time in the output header. This allows you to know how recent the data in the output file is. In addition, when generating area reports, RMU includes the name of the database in the report header.

- Extensible after-image journal files can be renamed

  Previously, the Rename qualifier for the RMU Backup After_Journal command was valid only for fixed-size .aij files. Beginning with this release of Oracle Rdb, you can use the Rename qualifier with extensible .aij files. See the RMU Backup After_Journal Command for details.

- Enhanced method for dumping portions of the .aij file

The Start and End qualifiers for the RMU Dump After_Journal command are difficult to use because you seldom know, nor can you determine, the AIJ record number prior to issuing the command. Two new qualifiers, First and Last, have been added to the RMU Dump After_Journal command to make dumping portions of the .aij file easier. See the RMU Dump After_Journal Command for more information.

- Wait qualifier added to RMU Open command

  In previous versions of Oracle Rdb, the RMU Open command could return the system prompt before a database was completely open and available. The new Wait qualifier allows you to specify that you do not want the system prompt returned until the database is completely open and available. See the RMU Open Command for more information.

- Allow rollback of a restore operation that performs a database conversion

  A new qualifier (Nocommit) has been added to the RMU Restore command. When you restore a backup file from a previous version of Oracle Rdb and specify the Nocommit qualifier, the database conversion is not permanent. You can roll back the restored database to its original version or permanently commit it to the current version at a later time. See the RMU Restore Command for details.

- Verification of disabled constraints

  You can now verify disabled constraints. You might use this to provide a business rule in the database that needs checking only occasionally. This is a useful practice if the overhead of checking the constraint during operating hours is too expensive, or if it is already being enforced by the application. See the RMU Verify Command for details.

- Enhancements to the RMU Show Statistics command

  Several enhancements have been made to the RMU Show Statistics command, as follows:

  – New screens to provide more information on classes of stalls

    See Section 6.1 for details.

  – Improved formatting of stall message strings

    See Section 6.3 for details.

  – New screens to display logical area statistics

    See Section 6.2 for details.

– New qualifier to reduce virtual memory reserved in anticipation of displaying Logical Area screens

  See Section 6.2.1 and the RMU Show Statistics Command for details.

– Ability to log all lock timeout and lock deadlock messages

  See Section 6.4 for details.

– Ability to signal events and specify the actions to be performed when those events occur

  See Section 6.5 for details.

– Ability to import and export a configuration file that specifies RMU Show Statistics qualifier settings and display settings

  See Section 6.6 for details.

– Ability to get real-time collection and presentation of database statistics from other nodes in the cluster where the database is currently being accessed

  See Section 6.7 for details.

– Screen header region indicates if clusterwide statistics are being collected

  See Section 6.7.1 for details.

– Ability to automatically capture images of all screens at a specified interval

  See Section 6.8 for details.

– Ability to display information about each AIJ backup operation being performed on the current node

  See Section 6.9 for details.

– CPU time is now displayed correctly

  See the RMU Show Statistics Command for details.

– The RMU Show Statistics command is integrated with the RMU Dump command to provide run-time database page information

  See Section 6.10 for details.

– Virtual memory statistics are no longer collected or displayed

  See the RMU Show Statistics Command for details.

- Ability to estimate the time it will take to roll back a transaction or to completely recover a failed process

  See Section 6.11 for details.

- The sorted File IO Overview and File Lock Overview screens provide additional sorting and filtering capabilities

  See Section 6.12 for details.

- Ability to specify whether you want to confirm before exiting the Show Statistics display

  See the RMU Show Statistics Command for details.

- Ability to display fast incremental backup run-time statistics

  See Section 6.13 for details.

- Ability to display hot standby recovery statistics

  See Section 6.14 for details.

- Ability to display statistics that summarize database transaction activity and transaction and verb execution rates

  See Section 6.15 for details.

- Ability to report a stall message at each interval

  See Section 6.1.4 for details.

- Reason for fetching a SPAM page now displayed

  See Section 6.16 for details.

- Ability to set a timeout value for user prompts

  See the RMU Show Statistics Command for details.

■ Elapsed time shown in Show commands

The RMU Show System and RMU Show Users commands now display elapsed as well as absolute times for the time that the monitor was started and the time that databases were opened.

The following example demonstrates this output:

```
$ RMU/SHOW USERS
Oracle Rdb V7.1-0 on node HOTRDB  2-APR-1998 16:56:05.43
    - monitor started  1-APR-1998 16:51:09.37 (uptime 1 00:04:56.06)
    - monitor log filename is "DISK$:[RDM$MONITOR]RDMMON70.LOG;1"
```

```
database DISK$:[DB]MYDB.RDB;1
    - first opened  2-APR-1998 16:56:04.85 (elapsed 0 00:00:00.59)
    - 1 active database user
    - 22E07174:1 - BATCH_874 - non-utility, RDBTESTER - active user
        - image DISK$:[RDBVMS]RDBTESTER.EXE;1
```

1.4 New and Changed Features for Hot Standby

The new and changed features for hot standby are described in the following subsections.

1.4.1 Status of Hot Standby Replication Governor Is Database Attribute

The status of the Hot Standby Replication Governor is now a database attribute. The RMU Replicate After_Journal Configure command supports the enabling and disabling of the Replication Governor. The RMU Dump Header command can be used to view the status of the Replication Governor for a given replication standby database.

The following example shows the RMU Replicate After_Journal Configure command.

```
$ RMU/REPLICATE AFTER_JOURNAL CONFIGURE -
_$ USER_DISK1:[ORACLE_RDB.STANDBY]MF_PERSONNEL -
_$  /MASTER_ROOT=REMOTE1::USER_DISK2:[ORACLE_RDB.MASTER]MF_PERSONNEL.RDB -
_$ /GOVERNOR=ENABLED
```

The following example shows the output of an RMU Dump command with the Header=Hot_Standby qualifier:

```
*------------------------------------------------------------------------
* Oracle Rdb V7.1-00                                3-DEC-1996 10:48:02.68
*
* Dump of Database header
*   Database: USER_DISK1:[ORACLE_RDB.STANDBY]MF_PERSONNEL.RDB;1
*
*------------------------------------------------------------------------
Database Parameters:
    Root filename is "USER_DISK1:[ORACLE_RDB.STANDBY]MF_PERSONNEL.RDB;1"
    Hot Standby...
     - Database has been configured as "Replication Standby"
         Master database is "USER_DISK2:[ORACLE_RDB.MASTER]MF_PERSONNEL.RDB;1"
         Remote node name is "REMOTE1"
```

```
Replication last commenced on 21-NOV-1996 09:03:36.92
Database replication is "online"
Replication Governor is "enabled"
Server checkpoint interval is 100 messages
Server gap-timeout interval is 5 minutes
Server buffer count is 256
Server 2PC transaction resolution is "commit"
```

See the *Oracle Rdb7 and Oracle CODASYL DBMS Guide to Hot Standby Databases* for complete information on hot standby databases.

1.4.2 Providing the Master Database with an Alternate Node Name

The RMU Replicate After_Journal command has been enhanced to support a new optional qualifier, Alt_Remote_Node=node-name, that allows you to identify an available secondary network link to the standby database through an alternate remote node name. The purpose of the alternate remote node name is to provide the master database with uninterrupted hot standby replication if the network fails (and if multiple network links are available). Under these circumstances, the master database will automatically reconnect to the standby database using the alternate remote node name information.

You can only use the Alt_Remote_Node qualifier with the Standby_Root qualifier. The maximum length of the node name is 31 characters. The node name can be the same as the node name specified with the Standby_Root qualifier. The node name specified with the Alt_Remote_Node qualifier must identify the same standby database on the same remote node as you originally specified with the Standby_Root qualifier.

If you do not specify the Alt_Remote_Node qualifier and a network failure occurs, the master database automatically attempts to reconnect to the standby database using the original remote node name that you specified with the Standby_Root qualifier.

You can use the RMU Replicate After_Journal Configure command to store the alternate remote node name in the database and the RMU Replicate After_Journal Reset command to clear any previously configured alternate remote node name information.

At run time, you can define the RDM$BIND_HOT_NETWORK_ALT_NODE logical name in the LNM$SYSTEM_TABLE table to override any alternate remote node name information specified when hot standby was started. The logical name must be specified on all nodes where the master database is open.

1.4.3 Row Caching Not Allowed While Hot Standby Replication Is Active

Row caching cannot be enabled on a hot standby database while replication is active. The hot standby feature will not start if row caching is enabled.

This restriction exists because rows in the row cache are accessed through logical dbkeys. However, information transferred to the hot standby database through the after-image journal facility only contains physical dbkeys. Because there is no way to maintain rows in the cache through the hot standby processing, the row cache must be disabled on the standby database when the standby database is open and replication is active. The master database is not affected; the row cache feature and the hot standby feature may be used together on a master database.

To open a hot standby database prior to starting replication, use the RMU Open command with the Row_Cache=Disabled qualifier.

1.5 New SQL*Net for Oracle Rdb Release 7.1.5

This release of Oracle Rdb includes SQL*Net for Oracle Rdb release 7.1.5.

SQL*Net for Oracle Rdb release 7.1.5 allows you to access and manage data in Oracle Rdb databases from applications written using Oracle Developer (or those using OCI or PL/SQL interfaces).

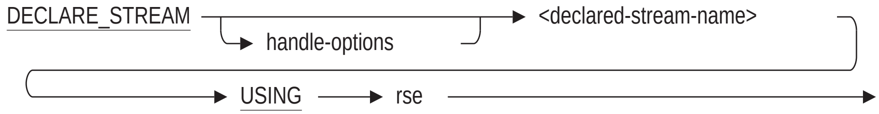
SQL*Net for Oracle Rdb is installed as a part of Oracle SQL/Services release 7.1.5. See the *Oracle SQL/Services Release 7.1.5 Installation Guide* for complete information and installation instructions.

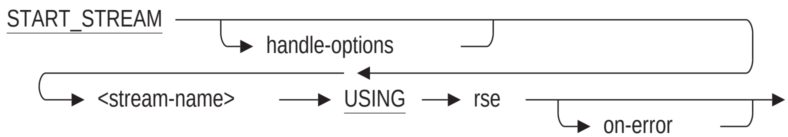
1.6 New and Changed Features for RDO, RDBPRE, RDML, and RDB$INTERPRET

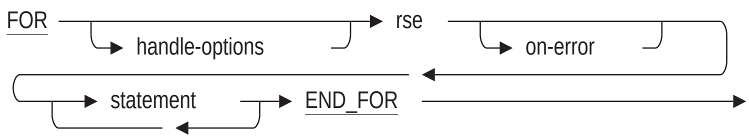
This section describes new and changed features for RDO, RDBPRE, RDML, and RDB$INTERPRET. The new features are two new keywords that have been added to the handle-options for the DECLARE_STREAM, START_STREAM (undeclared format), and FOR loop statements for RDBPRE, RDO, RDML, and RDB$INTERPRET.

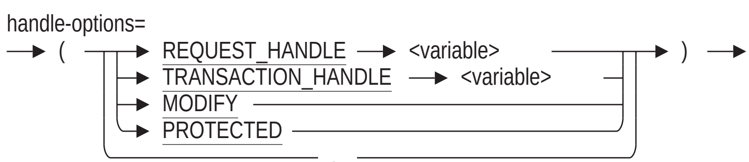
In prior releases, the handle-options could not be specified in interactive RDO or RDB$INTERPRET. This is no longer true. However, these allowed handle options are limited to the MODIFY and PROTECTED keywords. For RDBPRE, all handle-options listed are supported. These option names are existing keywords; no new keywords have been added to the RDO language.

The enhanced statements are shown in the following diagrams:









The following list describes the new options that are available for the handle-options clause:

- REQUEST_HANDLE

  Specifies the request handle for this request. This option is valid for RDBPRE and RDML applications only. It cannot be used with RDB$INTERPRET or interactive RDO.

- TRANSACTION_HANDLE

Specifies the transaction handle under which this request executes. This option is valid for RDBPRE and RDML applications only. It cannot be used with RDB$INTERPRET or interactive RDO.

- MODIFY

  Specifies that the application will modify all (or most) of the records fetched from the stream or FOR loop. You can use this option to improve application performance by avoiding lock promotion from SHARED READ access for the FETCH statement to PROTECTED WRITE access for the nested MODIFY or ERASE statement. It can also reduce deadlocks because lock promotions are avoided.

  This option is valid for RDBPRE, RDB$INTERPRET, and interactive RDO. This option is not valid for RDML.

  For example:

```
RDO>   FOR (MODIFY) E IN EMPLOYEES WITH E.EMPLOYEE_ID = "00164"
cont>    MODIFY E USING E.MIDDLE_INITIAL = "M"
cont>    END_MODIFY
cont>  END_FOR
```

  This FOR loop statement uses the MODIFY option to indicate that the nested MODIFY is an unconditional statement and, therefore, aggressive locking can be undertaken during the fetch of the record in the FOR loop.

- PROTECTED

  Specifies that the application can modify records fetched by this stream with a separate and independent MODIFY statement. Therefore, this stream should be protected from interference (known as the Halloween effect). The optimizer selects a snapshot of the rows and stores them in a temporary relation for processing, rather than traversing indexes at the time of the FETCH statement. In some cases, this might result in poorer performance when the temporary relation is large and overflows from virtual memory to a temporary disk file, but the record stream will be protected from interference. See the *Oracle Rdb7 Guide to Database Performance and Tuning* for information about the Oracle Rdb logical names RDMS$BIND_WORK_VM and RDMS$BIND_WORK_FILE.

  This option is valid for RDBPRE, RDB$INTERPRET, and interactive RDO. This option is not valid for RDML.

  The following example creates a record stream in a BASIC program using Callable RDO. In this example, the FETCH needs to be protected against MODIFY statements that execute in other parts of the application.

```
RDMS_STATUS = RDB$INTERPRET ('INVOKE DATABASE PATHNAME "PERSONNEL"')
RDMS_STATUS = RDB$INTERPRET ('START_STREAM (PROTECTED) EMP USING ' + &
                             'E IN EMPLOYEES')
RDMS_STATUS = RDB$INTERPRET ('FETCH EMP')


DML_STRING = 'GET ' +                                             &
                 '!VAL = E.EMPLOYEE_ID;' +                        &
                 '!VAL = E.LAST_NAME;' +                          &
                 '!VAL = E.FIRST_NAME' +                          &
             'END_GET'
RDMS_STATUS = RDB$INTERPRET (DML_STRING, EMP_ID, LAST_NAME, FIRST_NAME)
```

2

New Features Affecting All Interfaces

This chapter describes the new and changed features that affect all Oracle Rdb release 7.1 interfaces.

2.1 Information Tables

Information tables display internal information about storage areas, after-image journals, row caches, database users, the database root, and database character sets. Once the information tables are created, you can use the INFO_TABLES.SQL script to query them with the SQL interface.

Information tables are special read-only tables that can be created in an Oracle Rdb release 7.1 database and used to retrieve database attributes that are not stored in the existing relational tables. Information tables allow interesting database information, which is currently stored in an internal format, to be displayed as a relational table.

The script, INFO_TABLES.SQL, is supplied as a part of the Oracle Rdb kit. The INFO_TABLES.SQL file is in the SQL$SAMPLE directory. The file is copied to the SQL$SAMPLES directory if you install the sample files during installation.

For Oracle Rdb release 7.1, the following information tables are supported:

Table Name	Description
RDB$STORAGE_AREAS	Displays information about the database storage areas.
RDB$DATABASE_JOURNAL	Displays information about the default journal.
RDB$CACHES	Displays information about the database row caches.
RDB$DATABASE_ROOT	Displays information about the database root.
RDB$JOURNALS	Displays information about the database journal files.

Table Name	Description
RDB$DATABASE_USERS	Displays information about the database users.
RDB$LOGICAL_AREAS	Displays information about the logical areas.
RDB$CHARACTER_SETS	Displays information about the Oracle Rdb character sets.
RDB$NLS_CHARACTER_SETS	Displays the mapping of Oracle NLS character sets to Oracle Rdb character sets.

For additional information about these information tables on OpenVMS, see the ORACLE_RDB*nn* topic and select the Information_Tables subtopic (where *nn* is the version number if using multiversion) in the Oracle Rdb command-line Help.

The following example shows how to query one of the information tables created by the INFO_TABLES.SQL script.

```
SQL> SELECT * FROM RDB$LOGICAL_AREAS WHERE RDB$LOGICAL_AREA_NAME='JOBS';
 RDB$LOGICAL_AREA_ID   RDB$AREA_ID   RDB$RECORD_LENGTH   RDB$THRESHOLD1_PERCENT
   RDB$THRESHOLD2_PERCENT   RDB$THRESHOLD3_PERCENT   RDB$ORDERED_HASH_OFFSET
      RDB$RECORD_TYPE   RDB$LOGICAL_AREA_NAME
                   95           7                  41                          0
                         0                         0                          0
                     1   JOBS

1 row selected
```

2.1.1 Restrictions for Information Tables

- You cannot alter the information tables. The table names and column names must remain unchanged.
- Documentation on what each bit in the flag fields represents is available on OpenVMS. See the ORACLE_RDB*nn* topic and select the Information_Tables subtopic (where *nn* is the version number if using multiversion) in the Rdb command-line Help.

2.2 New Logical Names

The following sections describe new Oracle Rdb logical names for OpenVMS. Additional logical name descriptions appear in Section 1.1.

2.2.1 Controlling How Oracle Rdb Initializes File Contents

In certain cases, problems with the operating system file erase function have caused problems with the ability of Oracle Rdb to correctly initialize file contents with specific patterns.

To work around these problems, a new logical name, RDM$BIND_FORCED_DISK_ERASE, can be used to control how Oracle Rdb attempts to initialize files with specific patterns. If RDM$BIND_FORCED_DISK_ERASE is undefined or defined as zero, Oracle Rdb uses the operating system erase functionality to initialize file contents. If RDM$BIND_FORCED_DISK_ERASE is defined as 1, then Oracle Rdb initializes the file contents by writing a pattern into the file directly. Because writing to the file directly is slower than using the operating system erase features, it is not the default.

To be effective, the logical name RDM$BIND_FORCED_DISK_ERASE must be defined with the /SYSTEM qualifier before a database is opened or created.

2.2.2 Specifying the Size of the Hash Table for Temporary Tables

The RDMS$TTB_HASH_SIZE logical name specifies the size of the hash table used by temporary tables. If the logical name has not been defined, a default value of 1249 is used. If you expect that a temporary table will have 10,000 or more rows, use this logical name to adjust the hash table size so as to avoid long hash chains. Set this logical name to roughly one quarter of the expected maximum number of rows per temporary table. For example, if it is likely that a temporary table will be populated with 100,000 rows, then define this logical name to be 25,000. If there are memory constraints, you should define the logical name value to be no larger than 25,000.

2.2.3 Logical Name to Cause Operator Notification of Storage Area Extension

On OpenVMS platforms, the system operator classes CLUSTER and CENTRAL may be optionally notified for storage area extend events.

By default, notification of area extend events is disabled. To enable this notification, define the logical name RDM$BIND_NOTIFY_STORAGE_AREA_EXTEND to 1. This results in storage area extend events being notified. There is, however, no way to alter the operator classes that are notified.

The logical name RDM$BIND_NOTIFY_STORAGE_AREA_EXTEND should be defined systemwide, so that any process extending an area causes the notification to occur.

Following is an example operator message:

```
%%%%%%%%%%%  OPCOM  16-JAN-1997 05:44:11.71  %%%%%%%%%%%
Message from user SYSTEM on SMILES
Oracle Rdb Database DUA0:[DB]DB.RDB;1 Event Notification
Storage area DUA0:[DB]DB1.RDA;1 extending by 4321 disk blocks
```

2.2.4 Logical Name to Set Debug Flags

You can now use a logical name to set debug flags. The RDMS$SET_FLAGS logical name accepts a string in the same format as provided to the SQL SET FLAGS statement. Abbreviations, values, and negation (NO) of keywords are supported. The equivalence string is processed after the logical name during a database attach operation. Therefore, settings made with the RDMS$DEBUG_FLAGS logical name are superseded by keywords defined with this new logical name. Unlike other Oracle Rdb logical names, an exception is raised if an error is found in the equivalence string, and the attach to the database fails. The SQL SHOW FLAGS statement will display settings made with the RDMS$SET_FLAGS logical name.

See the SET FLAGS Statement for details on the format and arguments for the equivalence string.

2.2.5 Directing RUJ Information to a Global Section

For row caches, recovery-unit journaling (RUJ) must logically come before each modification to any record residing in a row cache. Having the RUJ information is critical in returning the row to its before-image state if the modifying transaction rolls back or terminates abnormally. To minimize the occurrences of these synchronous RUJ I/Os, Oracle Rdb defers writing, for as long as possible, the modified records into the row cache. The synchronous I/O includes all updated rows since the previous RUJ I/O.

If an application performs a large number of inserts or updates to a table contained in a row cache, a high number of these RUJ I/Os may be seen. To eliminate the majority of these RUJ I/Os, you can use the system logical name, RDM$BIND_RUJ_GLOBAL_SECTION_ENABLED, to specify whether you want the before-image records to be written to process-private memory (the traditional method) or to a systemwide, shared memory, global section.

When you choose the global section option, the RUJ information is made available to any possible future database recovery process from the shared memory global section. Traditionally, such information was only shared by writing the information to the .ruj file, which the DBR process could read. By adding this capability, only an in-memory I/O, not a synchronous I/O, is required before modifying a row in the row cache.

When a process terminates abnormally, Oracle Rdb activates a database recovery (DBR) process to recover the work done by the terminated user. The DBR process performs a rollback of the outstanding, uncommitted process transactions, if any. If the systemwide DBR process buffers are enabled, the DBR process first writes the current RUJ buffer to the .ruj file. It then recovers the .ruj file by placing the before-image of each record back on the database page. If the dbkey for that record is also found in a row cache, the before-image is placed back into the row cache as well.

To enable this optimization, define the logical name RDM$BIND_RUJ_GLOBAL_SECTION_ENABLED as 1 in the system logical name table. The global section created for the RUJ buffers will be about 16 OpenVMS Alpha pages for each allowed user of a database. One global section will be created for each database that has row cache enabled. Databases that do not have row cache enabled will not have the RUJ global buffer optimization enabled.

The OpenVMS system parameter GBLSECTIONS will need to be increased by the maximum number of Oracle Rdb databases open at one time on the system. The OpenVMS system parameter GBLPAGES will need to be increased by 256 times the maximum number of users for each database open at one time on the system. In addition, the OpenVMS system parameter GBLPAGFIL will need to be increased by 16 (on OpenVMS Alpha) times the maximum number of users for each database open at one time on the system.

There is no additional virtual memory consumption for database user processes when the RUJ global buffers optimization is enabled; each user process continues to use the same amount of virtual memory (256 blocks) as when the optimization is not enabled.

2.2.6 Limiting the Number and Size of AIJ Initialization I/O Buffers

When an AIJ backup operation completes, the after-image journal files are initialized with a pattern of -1 (hex FF) bytes. This initialization is designed to be as fast as possible and thus fully use the I/O subsystem by performing many large, asynchronous I/Os at once. This speed can, however, come at the cost of a high load on I/O components during the initialization. This load could slow down other I/Os on the system.

To allow control over the relative I/O load that the AIJ initialization operation places on the system, two logical names have been created. On OpenVMS, these logical names should be defined in the system logical name table and are translated each time an .aij file is initialized.

The RDM$BIND_AIJ_INITIALIZE_IO_COUNT logical name specifies the number of asynchronous I/O operations that will be queued at once to the .aij file. If the logical name is not defined, the default value is 15. The minimum value is 1 and the maximum value is 32.

The RDM$BIND_AIJ_INITIALIZE_IO_SIZE logical name controls the number of 512-byte disk blocks to be written per I/O. If the logical name is not defined, the default is 127. The minimum value is 4 and the maximum value is 127.

Reducing the value of either logical name is likely to increase the amount of time needed to initialize the .aij file after a backup. However, it may also reduce load on the I/O subsystem.

2.2.7 Operator Notification of Bugchecks

On OpenVMS, when an Oracle Rdb process writes a bugcheck dump file, the system operator classes CLUSTER and CENTRAL are notified. The operator message includes the Oracle Rdb release number, the user name, the process ID, and the bugcheck dump file name.

To disable this notification, define the logical name RDM$BUGCHECK_IGNORE_FLAGS to include "O" in the translation. This results in the previous behavior (no operator notification). There is, however, no way to alter the operator classes that are notified.

Following is an example operator message:

```
%%%%%%%%%%%  OPCOM  15-JAN-1997 04:55:11.78  %%%%%%%%%%%
Message from user SYSTEM on RDBNT
Oracle Rdb Database DUA0:[DB]DB.RDB;1 Event Notification
Process 00000EFA generating bugcheck dump file DUA0:[DB]RDSBUGCHK.DMP;
```

2.2.8 Disabling the RUJ and AIJ During Certain Operations

You can use the logical name RDMS$CREATE_LAREA_NOLOGGING or the statement keyword NOLOGGING to disable journaling to the recovery-unit journal (.ruj) file and after-image journal (.aij) file during certain CREATE and ALTER operations.

Usually, when you create a new logical area as part of a CREATE TABLE, CREATE STORAGE MAP, CREATE INDEX, ALTER STORAGE MAP, or ALTER INDEX statement, all updates to these logical areas are journaled to the recovery-unit and after- image journals.

This can be a problem when you create or alter a large index or reorganize a storage map for a large table. In these cases, table rows, hash buckets, or B-tree nodes are written to the new logical areas and must be journaled to the .ruj and .aij files. As the data definition language (DDL) operation proceeds, these records might also be journaled again due to a subsequent update. The amount of I/O to these journals might be extensive, and the long duration of the transaction might cause the .ruj and .aij files to grow quite large.

By using the RDMS$CREATE_LAREA_NOLOGGING logical name or the NOLOGGING keyword when you issue the statement, the I/O to the recovery and after-image journals can be almost eliminated. The recovery and after-image journals will contain only a special logical operation with no associated data for the CREATE or ALTER operation. These DDL operations reduce journaling I/O and lower disk space requirements for these operations on large tables.

Note: Be aware of the possible disadvantages of disabling journaling. The trade-off is less I/O during the operation versus more complex recovery procedures. The changes in recovery are discussed in the following sections. These sections provide information on using this logical name. The issues described in the following sections also apply to the use of the NOLOGGING keyword with SQL statements.

2.2.8.1 Disabling and Enabling Logging

To disable journaling for the CREATE TABLE, CREATE STORAGE MAP, CREATE INDEX, ALTER STORAGE MAP, or ALTER INDEX statements, you can define the logical name RDMS$CREATE_LAREA_NOLOGGING for the process that will perform these operations. Once you attach to the database, all journaling of new logical areas is disabled until the transaction that created them has been committed or rolled back.

The values accepted for this logical name are as follows:

- 0 - Specifies that journaling is enabled when a logical area is created. This is the default if the logical name is not defined.

  1 - Specifies that logical area journaling is disabled when a logical area is created.

On OpenVMS, the logical name is defined as shown in the following example:

```
$ DEFINE RDMS$CREATE_LAREA_NOLOGGING "1"
```

2.2.8.2 Database Recovery with Enabled AIJ and Disabled Logging

If after-image journaling is enabled when the RDMS$CREATE_LAREA_NOLOGGING logical name is in effect, then a warning message is issued to inform you that journaling was disabled during the transaction. The message recommends that a full database backup be performed as soon as possible, because the after-image journal file no longer contains all changes required to rebuild the new or altered database object. In effect, a hole or gap exists, which means that the database cannot be fully recoverable from the after-image journal files.

```
$ DEFINE/USER RDMS$CREATE_LAREA_NOLOGGING 1
$ SQL$
SQL> ATTACH 'FILE DB$:TEST_NOJOURNAL';
SQL> CREATE INDEX T_I ON T (A);
%RDB-W-META_WARN, metadata successfully updated with the reported warning
-RDMS-W-DATACMIT, unjournaled changes made; database may not be recoverable
-RDMS-W-DOFULLBCK, full database backup should be done to ensure
 future recovery
SQL> COMMIT;
```

When an after-image journal file contains a record of an unjournaled CREATE or ALTER statement and is used to roll forward the after-image journal using the RMU Recover command, then the logical area is marked to indicate that it is corrupt. This can be seen using the RMU Dump Larea=RDB$AIP command:

```
                      0003 0037  014F  logical area 55, physical area 3
                             06  0153  area name length 6 bytes
000000000000000000454C504D4953  0154  area name 'SIMPLE.........'
  000000000000000000000000000000  0164  area name '...............'
                       00000000  0173  snaps enabled TSN 0
                           000D  0177  record length 13 bytes
                       00000000  0179  MBZ '....'
                             0D  017D  entry is in use
                           0000  017E  MBZ '..'
                         000000  0180  thresholds are (0,0,0)
                             00  0183  MBZ '.'
                             area is corrupt and cannot be accessed
```

Oracle Rdb reports that the table or index is corrupt (has had unjournaled changes) if an attempt is made to use that table or index after the recovery is complete. The only option now is to drop the table, storage map, or index and repeat the operation. In the following example, the index T_I was created with logging disabled, and the RMU Recover command was used to recover the database.

```
SQL> SET FLAGS 'STRATEGY';
SQL> -- If you select using the incomplete index,
SQL> -- the query fails.
SQL> SELECT * FROM T WHERE A > 0;
Leaf#01 FFirst T Card=5
  BgrNdx1 T_I [1:0] Fan=17
%RDMS-F-DATATBLCMIT, unjournaled changes made to user-defined object
SQL>
SQL> -- Now do a sequential scan. This succeeds.
SQL> SELECT B FROM T;
           B
        NULL
           1
        NULL
           1
           2
5 rows selected
SQL>
SQL> -- Now drop the index.  The operation succeeds.
SQL> DROP INDEX T_I;
Firstn  Get     Retrieval by index of relation RDB$INDICES
  Index name  RDB$NDX_NDX_NAME_NDX [1:1]        Direct lookup
SQL>
SQL> -- Select again (uses sequential scan).
SQL> -- The query succeeds.
SQL> SELECT * FROM T WHERE A > 0;
           A            B
           1         NULL
           1            1
           2         NULL
           2            1
           2            2
5 rows selected
SQL>
SQL> -- Re-create the index.
SQL> CREATE INDEX T_I ON T (A);
SQL>
SQL> -- Select using the new index.
SQL> -- The operation succeeds.
SQL> SELECT * FROM T WHERE A > 0;
Leaf#01 FFirst T Card=5
  BgrNdx1 T_I [1:0] Fan=17
```

```
           A            B
           1         NULL
           1            1
           2         NULL
           2            1
           2            2
5 rows selected
SQL>
SQL> COMMIT;
```

Note: If a CREATE INDEX or ALTER INDEX statement refers to a hashed index, then the operation requires updates to the RDB$SYSTEM_RECORD on each page of a mixed format area. When these records are updated, the changes are always journaled to the recovery and after-image journal files. Therefore, some journaling activity results from these operations.

2.2.8.3 Database Recovery with Disabled AIJ and Disabled Logging

If after-image journaling is not currently enabled, then the rollback operation can fully recover the database from the recovery-unit journal file. Therefore, only the DATACMIT warning is issued in this case. This is a warning that an error reported during the transaction must be rolled back to guarantee recovery.

2.2.8.4 Report from RMU Verify When a Logical Area Is Marked Incomplete

If a logical area is marked incomplete, the RMU Verify command attempts to ready the incomplete logical area. The following example shows that the index T_I is incomplete (the warning message DATATBLCMIT is returned) and the verification of the B-tree is abandoned.

```
$ RMU/VERIFY/ALL db$:test_nojournal
.
.
.
%RMU-I-BGNNDXVER, beginning verification of index T_I
%RMU-W-DATATBLCMIT, unjournaled changes made to user-defined object
%RMU-E-BDLAREADY, error readying logical area with dbid 48
%RMU-W-NOT_LARDY, area for 48:560:0 not in proper ready mode
%RMU-E-BADDBKFET, Error fetching dbkey 48:560:0
%RMU-W-BTRVFYPRU, B-tree verification pruned at this dbkey
%RMU-I-BTRROODBK, root dbkey of B-tree is 48:560:0
%RMU-I-NDXERRORS,        2 index errors encountered
```

```
%RMU-I-ENDNDXVER, completed verification of index T_I
.
.
.
```

2.2.8.5 Effect of CREATE or ALTER Statement Failure with Journaling Disabled

If a CREATE or ALTER statement fails when journaling is disabled, the logical area creation (which is always journaled) is rolled back. All data written to that logical area is erased from the database. To erase the data from a mixed format area requires that each page of the storage area be processed, and so will most likely be slower than similar recovery when journaling is enabled.

When recovery is performed using the RMU Recover command, any rolled back transaction is discarded (is not applied to the backup database) so that no reference to the incomplete logical area is encountered.

2.2.8.6 Effect of an Error During the Transaction When Journaling Is Disabled

If the transaction that performed the CREATE or ALTER statement has already been committed, then subsequent transactions will have resumed journaling. This is the normal logging mode for Oracle Rdb, and errors will be handled as expected.

However, if the original transaction that performed the CREATE or ALTER statement is still active and an error occurs while writing to an unjournaled logical area, then the logical area is immediately marked as corrupt. Such errors include failures of INSERT or UPDATE statements due to duplicate key values, constraint violation, and database locking errors. The transaction should be aborted using the ROLLBACK statement.

Although you can use the COMMIT statement and you can delete the logical area (using a DROP statement), this action may leave data anomalies that could not be rolled back at the time the error was detected. Oracle Corporation recommends that you roll back the transaction.

Oracle Corporation recommends that you commit the transaction immediately after the CREATE or ALTER statement has successfully completed. That is, avoid performing additional data manipulation statements (DML) statements such as INSERT, UPDATE, and DELETE. By committing promptly, you avoid the problems described in this section. This action also releases locks on rows and other database system resources.

2.2.8.7 Effect on Databases Enabled for Hot Standby

Hot standby requires all operations to be journaled; therefore, the RDMS$CREATE_LAREA_NOLOGGING logical name is ignored by any database enabled for hot standby.

2.2.8.8 Restriction for LIST Storage Map

Disabling logging is not supported when you create or alter a LIST storage map. The RDMS$CREATE_LAREA_NOLOGGING logical name is ignored when you issue statements to create or alter a LIST storage map.

The reason for this restriction is that it is not possible for the rollback processing to distinguish between old and new LIST segments that might exist in the storage map.

2.3 Unique Identification in Bugcheck Dump Files

Oracle Rdb now includes a unique identifier in bugcheck dump files. This unique identifier is intended to allow bugcheck dumps to be more easily identified and possibly cataloged by users and by Oracle Corporation. This identifier is a 32-byte character string and is generated using the system boot time, the time of the bugcheck dump, and the process identification of the process writing the dump.

The following is an example of the unique identifier in a bugcheck dump file:

```
====================================================================
          Bugcheck dump FSA0:[DB]RDSBUGCHK.DMP;
====================================================================

This file was generated by Oracle Rdb V7.1 - upon detection of a fatal,
unexpected, error.  Please return this file, the query or program that produced
the bugcheck, the database, monitor log, and any other pertinent information to
your Oracle support representative for assistance.
This is a AlphaServer 2100 4/275 running VMS V7.2
Current time is 16-MAY-1998 11:08:14.25
Bugcheck Dump Identification is: "PS6J3WHQF4GBAA2UD32L8BPVA23PBAAA"
```

2.4 Optimizer Zigzag Strategy Now Uses a Temporary Relation

Oracle Rdb sometimes uses a temporary relation for storing intermediate results fetched from the inner loop of a join. This temporary relation avoids a rescan of the index when performing duplicate processing for the outer loop and results in lower I/O cost for zigzag match strategy.

The side effect is that Oracle Rdb now uses virtual memory for the temporary relation that can be controlled using the logical name RDMS$BIND_WORK_VM. If there are many duplicate values in the inner loop of the join, then the temporary relation may overflow into a temporary file on disk (the location of which is controlled by the logical name RDMS$BIND_WORK_FILE). These logical names are documented in the *Oracle Rdb7 Guide to Database Performance and Tuning*. To avoid the overflow from virtual memory to disk file, assign a larger byte count to the logical name RDMS$BIND_WORK_VM before you attach to the database. Making this value larger may allow queries to keep these temporary results in memory and reduce the physical I/O to disk.

You may need to increase the working set for the process if some queries use many zigzag strategies, because each creates a temporary relation. For instance, the following portion of a query strategy shows that the zigzag strategy is used for the match strategy on both the inner and outer loops. The inner loop manages the temporary relation.

```
Reduce  Sort
Cross block of 8 entries
  Cross block entry 1
    Conjunct
    Match
      Outer loop      (zig-zag)
        Conjunct
        Leaf#01 Sorted TABLE_A Card=11924
          FgrNdx  INDEX_1 [3:3] Bool Fan=5
          BgrNdx1 INDEX_2 [2:2] Fan=5
      Inner loop      (zig-zag)
        Conjunct        Get     Retrieval by index of relation TABLE_B
          Index name  INDEX_3 [2:2]
  Cross block entry 2
.
.
.
```

2.5 Process Image Name Written to Monitor Log on OpenVMS

For local (nonremote) database attach requests, the process image name (the main image being run by the process) is now logged to the Oracle Rdb monitor log file.

The following is an example of the image name display:

```
5-JUN-1997 12:43:03.49 - Received user attach request from 2020412F:1
  - process name A_TOLLIVER_1, user TOLLIVER
  - image name "DKA0:[SYS0.SYSEXE]SQL$.EXE;1"
  - for database "_$1$DKA0:[DB]DB.RDB;1" [_$1$DKA0] (738,510,0)
  - database global section name is "RDM70T_B4I0ATO0"
  - database global section size is 99 pages (512 bytes per page)
  - database dashboard installed
  - cluster watcher is active
  - sending normal user attach reply to 2020412F:1
```

If the image name is not available for remote attaches or because of unexpected error conditions, a message of "image name not available" is written to the log file in place of the process image name.

2.6 Monitor Processing of OPCOM Messages

When an Oracle Rdb process needs to send an OPCOM message to the system operator (to notify the operator that an AIJ switch operation has occurred, for example), the message may have to be queued for a delayed action by the sending process. This is the result of a restriction while calling the SYS$SNDOPR system service from executive mode while on an alternate stack.

To avoid the processing required to queue and later send the message when running in executive mode, processes now send the message to the Oracle Rdb monitor process (RDMMON) that will, in turn, call the SYS$SNDOPR system service to perform the operator notification. When in user mode, processes will call the SYS$SNDOPR system service directly (that is, the Oracle Rdb monitor process will not be requested to send the message).

When the Oracle Rdb monitor process is requested to perform the operator notification, it writes the message text to the monitor log file (RDMMON.LOG) as well as sending it to the system operator.

2.7 Monitor Process Quotas Increased

When an Oracle Rdb monitor process (RDMMON) is started using the RMU Monitor Start command, the quota limit that the monitor process uses is determined as the largest of the following three factors:

- A minimum value shown in Table 2–1

- The quota value from the user designated by the RDM$MON_USERNAME logical name (with a default value of "SYSTEM")
- The quota value from the process performing the startup operation

The minimum value of several of these quotas has been increased. The minimum value for each monitor quota is shown in Table 2–1.

Table 2–1 Monitor Process Minimum Quotas

Quota	Minimum Value
ASTLM	256
BIOLM	256
BYTLM	250000
DIOLM	256
ENQLM	32767
FILLM	2048
PGFLQUOTA	250000
PRCCNT	64
TQCNT	256
WSEXTENT	512
WSQUOTA	512

These quota value minimums are adjusted to help prevent the monitor from being unable to open numerous large databases.

2.8 Monitor Process Uses Less ENQLM

The Oracle Rdb monitor process holds null mode locks on a number of database resources to keep the lock value blocks valid even when no users are attached to a database. For systems that have a large number of databases or databases with a large number of storage areas, the monitor process can use many locks, sometimes exceeding its lock quota (ENQLM) even at the OpenVMS maximum value of 32767 locks.

The effect of this situation has been reduced.

By using the LCK$M_NOQUOTA flag when taking out many of these locks (in particular, the database FILID, SAC, RCACHE, TSNBLK, and SEQBLK locks), the monitor process uses less of its ENQLM. The total number of locks and system resources consumed remains the same, but the monitor process ENQLM is not deducted for these locks.

2.9 Calling Oracle Rdb from Shared Images

If code in the image initialization routine of a shared image makes any calls into Oracle Rdb through SQL or any other means, access violations or other unexpected behavior may occur if the Oracle Rdb images have not had a chance to do their own initialization.

To avoid this problem, applications must do one of the following things:

- Do not make Oracle Rdb calls from the initialization routines of shared images.
- Do link in such a way that the RDBSHR.EXE image initializes first.

  You can do this by placing the reference to RDBSHR.EXE and any other Oracle Rdb shared images last in the linker options file.

This is not a bug; it is a restriction resulting from the way OpenVMS image activation works.

2.10 Online Creation of Storage Areas Now Performed in Parallel

Similar to the CREATE DATABASE MULTITHREAD AREA ADDITIONS functionality, an online storage area addition now initializes the pages of multiple storage areas in a multithreaded, or parallel, operation. Multithreaded storage area initialization permits multiple I/O operations to be issued to multiple devices, which is likely to reduce the amount of time needed to create and initialize the storage areas.

In the following example, the 10 new storage areas are created on 10 different disk devices. Assuming adequate process quotas, the 10 areas (the 5 live storage areas as well as the 5 snapshot storage areas) will be initialized with parallel I/O. This reduces the overall time needed to initialize the storage areas.

```
SQL> ALTER DATABASE FILE MYDB
  ADD STORAGE AREA S1 FILENAME D1:[DB]S1 ALLOCATION 1000000
      SNAPSHOT FILENAME D6:[DB]S1 SNAPSHOT ALLOCATION 10000
  ADD STORAGE AREA S2 FILENAME D2:[DB]S2 ALLOCATION 1000000
      SNAPSHOT FILENAME D7:[DB]S2 SNAPSHOT ALLOCATION 10000
  ADD STORAGE AREA S3 FILENAME D3:[DB]S3 ALLOCATION 1000000
      SNAPSHOT FILENAME D8:[DB]S3 SNAPSHOT ALLOCATION 10000
  ADD STORAGE AREA S4 FILENAME D4:[DB]S4 ALLOCATION 1000000
```

```
    SNAPSHOT FILENAME D9:[DB]S4 SNAPSHOT ALLOCATION 10000
ADD STORAGE AREA S5 FILENAME D5:[DB]S5 ALLOCATION 1000000
    SNAPSHOT FILENAME D0:[DB]S5 SNAPSHOT ALLOCATION 10000;
```

The multithreaded online storage area addition feature is enabled by default. To disable multithreaded online storage area additions, define the logical name RDM$BIND_ONLINE_AREA_ADD_MULTITHREAD_COUNT to 0. Offline storage area addition does not use the multithreaded feature and continues to function as in previous versions of Oracle Rdb. Oracle Corporation recommends that you reserve storage area slots and then use online storage area addition.

By default, Oracle Rdb initializes up to 16 storage area files in parallel and issues up to 2 write I/O requests per storage area at a time. The logical name RDM$BIND_ONLINE_AREA_ADD_MULTITHREAD_COUNT can be used to limit the number of storage areas that are initialized in parallel. Define this logical name to a value less than 128 to limit the number of files being initialized at once.

Oracle Rdb attempts to limit the number of parallel operations based on the remaining process FILLM, ASTLM, and DIOLM quotas. To ensure the highest level of performance, specify values no less than the recommended minimums shown in Table 2–2 for these process and system quotas for online area additions.

Table 2–2 Recommended Quota Minimums for Online Storage Area Additions

Quota	Recommended Minimum
ASTLM	Two times the number of area files being added (including the snapshot storage area files), or 35, whichever is less
DIOLM	Two times the number of area files being added (including the snapshot storage area files) or 35, whichever is less
FILLM	At least enough available to open the additional number of storage area files being added (including the snapshot storage area files)
CHANNELCNT	At least enough available to open the additional number of storage area files being added (including the snapshot storage area files)
WSQUOTA	Large enough to avoid excessive page faulting. Each storage area being initialized in parallel requires at least an additional 25 working set pages on an OpenVMS Alpha system

In general, using more disk devices results in increased performance when you add multiple storage areas. If you specify a large number of storage areas and many areas share the same device, a large multithread count could cause excessive disk head movement, which could result in the storage area creation taking longer than if the areas were created one at a time. If this is the case, specify multiple ALTER DATABASE ... ADD STORAGE AREA statements or specify a smaller multithread count with the logical name RDM$BIND_ONLINE_AREA_ADD_MULTITHREAD_COUNT.

3

SQL New and Changed Features

This chapter describes the new and changed elements and statements for the SQL interface to Oracle Rdb release 7.1. It presents the information in reference format as it would appear in the *Oracle Rdb7 SQL Reference Manual*.

How to Read the Information Presented in This Chapter:

The information in this chapter does not describe all of the new and changed features for Oracle Rdb release 7.1. See the *Oracle Rdb Release Notes*, release 7.1, for descriptions of new and changed SQL features that were added but are not described in this chapter; this may include keywords, arguments, usage notes, syntax changes, and other features.

3.1 How to Read Syntax Diagrams

This section shows the format of SQL statements by using syntax diagrams. Syntax diagrams graphically portray optional, required, and repeating characteristics of SQL statements. You can learn the syntax of a statement by reading the syntax diagram for that statement.

To read a syntax diagram, start at the upper-left corner and follow the arrows until you exit from the diagram at the lower-right corner. When you come to a branch in the path, choose the branch that contains the option you want. If you want to omit an option, choose the path with no language elements. If a diagram occupies more than one horizontal line, the arrow returns below the end of each line to the left margin. Syntax diagrams can contain:

- Names of other syntax diagrams

If a diagram is named, the name appears in lowercase type above and to the left of the diagram. Syntax diagrams can refer to each other by name. The equal sign (=) indicates that the name is equivalent to the diagram and that the diagram can be substituted wherever that name appears in other diagrams. Such a substitution is similar to putting the name of a column where the syntax element column-name appears.

- Keywords

  Keywords appear in uppercase type. If a keyword is underlined, you must include it in the SQL statement. A keyword that is not underlined is optional. An optional keyword helps to make the statement more readable. Omitting or including an optional keyword has no effect on the statement.

- Punctuation marks

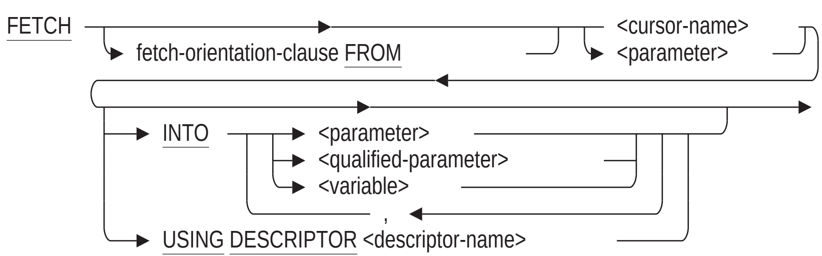
  Punctuation marks are included in the diagram when required by the syntax of the command or statement. For example, the semicolon (;) is a statement terminator in statements that require terminators.

- User-supplied elements

  User-supplied elements appear in lowercase type and within angle brackets (< >) in syntax diagrams. These elements can include names, expressions, and literals.

  If a user-supplied element appears on the main line of a diagram, as cursor-name does in Figure 3–1, you are required to supply a substitute for the element.

Figure 3–1 Sample Syntax Diagram (FETCH)



fetch-orientation-clause =

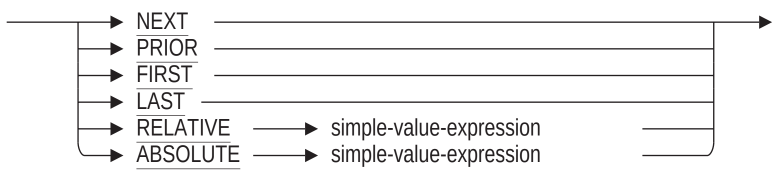


Figure 3–1 shows the syntax diagram for the SQL FETCH statement. Typically, the order of the descriptions of SQL statements follows syntax diagrams. The description is presented as an argument list, with each entry of the list describing the corresponding element of the syntax diagram. The following list shows the format of such an argument list but describes syntax diagram conventions instead of the meaning of each argument:

Arguments

FETCH

Is in uppercase type and underlined on the main line of the diagram. Therefore, you must supply this keyword.

fetch-orientation-clause FROM

Is in lowercase type and is below the main line of the diagram. Therefore, it is an optional clause. However, if you choose this clause, you must supply one of the keywords listed in the subdiagram along with the FROM keyword.

cursor-name
parameter

Is in lowercase type in angle brackets on the main line of the diagram. Therefore, you must supply a substitute for cursor-name or parameter.

INTO

Is optional; however, if you chose that branch, you must supply this keyword.

parameter
qualified-parameter
variable

Is in lowercase type in angle brackets on a branch. Because it always parallels an empty branch, parameters and variables are optional. The definitions for the common elements (such as parameter, qualified parameter, and variable shown in this example) appear in Chapter 2 of the *Oracle Rdb7 SQL Reference Manual*.

comma
Is on a reverse loop. The loop indicates that you have the option to include more than one parameter or variable. If you do, they are separated by commas.

3.2 Language and Syntax Elements

SQL uses a number of basic syntax and language elements that are common to many statements. These elements are sometimes referred to in syntax diagrams without further explanation. This chapter describes these new and changed elements:

- Character sets
- Data types
- Value expressions
- Predicates
- Select expressions and column select expressions

3.2.1 Supported Character Sets

Oracle Rdb supports multiple character sets and lets you use more than one character set in a database.

3.2.1.1 UNSPECIFIED Character Set

Oracle Rdb supports the use of the UNSPECIFIED character set. You can make comparisons and assignments between text objects (columns, literals, and so on) that have the UNSPECIFIED character set, and any other text object regardless of the character set of the other text object.

The characteristics of the UNSPECIFIED character set are as follows:

- The character set ID is 32767.
- It can be used to specify any session or database character set, including the identifier character set.
- It is a single-octet character set (fixed).
- It applies casing (uppercase and lowercase) only to ASCII characters.

- It contains ASCII, as follows:
  - The space character is the ASCII space character (0x20).
  - The wildcard character is the ASCII underscore (0x5f).
  - The string wildcard is the ASCII percent (0x25).

3.2.1.2 Oracle NLS Character Set Names

Oracle Rdb supports the use of Oracle National Language Support (NLS) names as aliases for existing Oracle Rdb character sets, as summarized in Table 3–1. You can use NLS alias names anywhere a character set name can be used.

Table 3–1 Oracle NLS Character Set Names Supported as Aliases

NLS ID	NLS Name	Alias for This Oracle Rdb Character Set
1	US7ASCII	ASCII
2	WE8DEC	DEC_MCS
31	WE8ISO8859P1	ISOLATIN1
35	CL8ISO8859P5	ISOLATINCYRILLIC
36	AR8ISO8859P6	ISOLATINARABIC
37	EL8ISO8859P7	ISOLATINGREEK
38	IW8ISO8859P8	ISOLATINHEBREW
41	TH8TISASCII	THAI
829	JA16VMS	DEC_KANJI
832	JA16SJIS	SHIFT_JIS
840	KO16KSC5601	KOREAN
850	ZHS16CGB231280	HANZI
865	ZH16BIG5	BIG5
1830	JA16EUCFIXED	KANJI

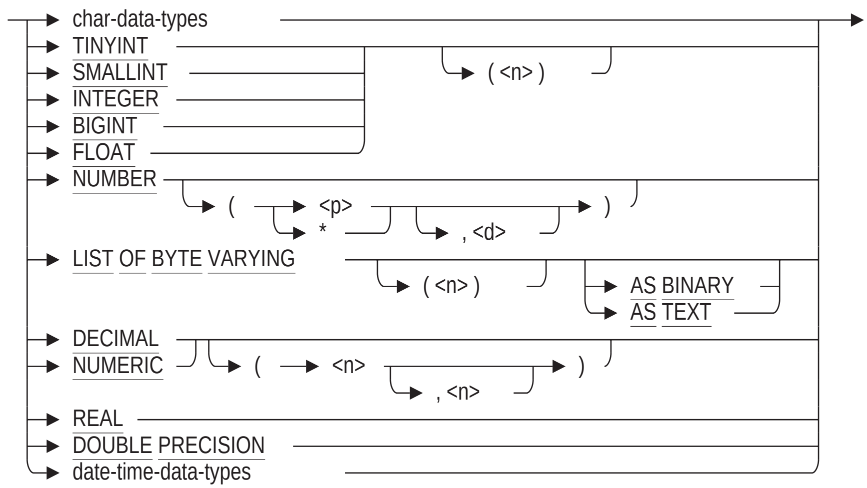
3.2.2 Data Types

When you define new columns of a table in the CREATE TABLE or ALTER TABLE statement, you must specify a data type for the column. The data type of a column controls how SQL interprets and stores values for that column. All value expressions (functions, parameters, and literals) have associated data types. See the

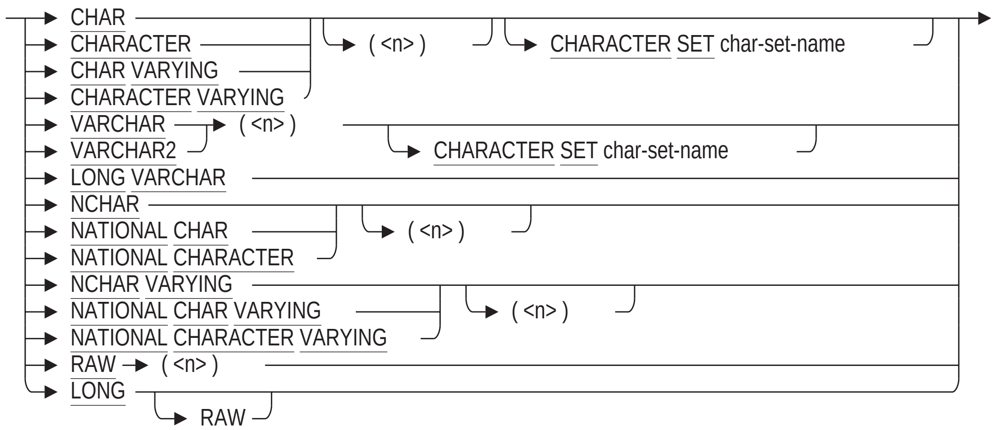
Oracle Rdb7 SQL Reference Manual for a comparison of SQL keywords and OpenVMS data types.

Use the following format when you specify a data type:

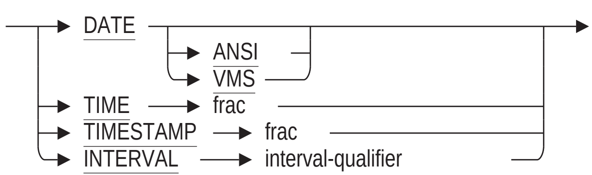
data-type =



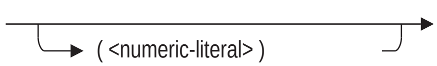
char-data-types =



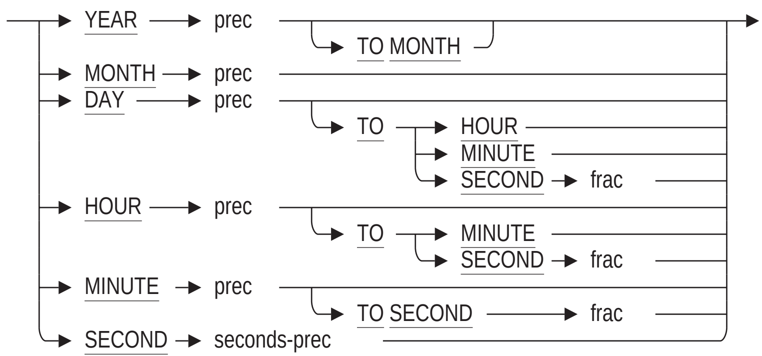
date-time-data-types =



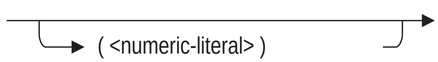
frac =



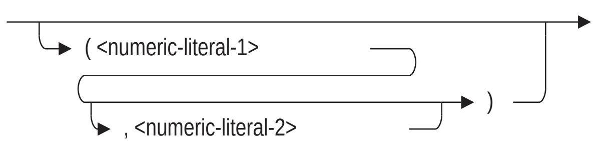
interval-qualifier =



prec =



seconds-prec =



3.2.2.1 Synonyms for Oracle Server Data Type Keywords

This release of Oracle Rdb adds support for several Oracle server data type keywords as synonyms for Oracle Rdb SQL keywords.

The following Oracle server data type keywords are synonyms for the specified Oracle Rdb data type keywords:

- VARCHAR2 is a synonym for VARCHAR.

- RAW is a synonym for VARCHAR. It always has a character set of UNSPECIFIED (that is, the CHARACTER SET clause is not permitted) and so can be assigned data from any other character set in the database. This allows data from any character set to be assigned to a column, parameter, or variable of this type. If you specify a length, the value can be 0 to 65535. Actual usage might be limited by available space in a table row, but the full length can be used by variable and parameter definitions (as is true for CHAR, VARCHAR, and VARCHAR2 data types).
- LONG is a synonym for LIST OF BYTE VARYING AS TEXT.
- LONG RAW is a synonym for LIST OF BYTE VARYING AS BINARY.

3.2.2.2 NUMBER Data Type

Oracle Rdb now supports the NUMBER data type. The NUMBER data type is added for compatibility with Oracle servers but has these differences: the Oracle server supports up to 38 digits of precision, and scale is restricted to between -84 and 127; Oracle Rdb release 7.1 supports 18 digits, and the scale is -128 to 127.

The following list highlights some of the features of the NUMBER data type as implemented for Oracle Rdb:

- NUMBER

  If no precision is provided, then this maps to DOUBLE PRECISION.

- NUMBER (p) or NUMBER (p, d), where p is precision and d is fractional precision

  When the precision (p) is specified, it is used to map to an integer type, or DOUBLE PRECISION if the precision is greater than 18.

  p <= 2, maps to TINYINT
  2 < p <= 4, maps to SMALLINT
  4 < p <= 9, maps to INTEGER
  9 < p <= 18, maps to BIGINT
  p > 18 maps to DOUBLE PRECISION

- NUMBER (*) or NUMBER (*, d)

  The asterisk (*) is shorthand for the largest scaled binary value. For Oracle Rdb release 7.1, this is equivalent to specifying 18 digits, and selects a BIGINT mapping.

- NUMBER(p, -d)

NUMBER allows d, the fractional precision, to be negative. If the scale is negative, the data is rounded to the specified number of places to the left of the decimal point. For example, a specification of (10,-2) means to round to hundreds.

Examples

```
SQL> CREATE TABLE A_TABLE (A NUMBER);
%SQL-I-NO_NUMBER, A is being converted from NUMBER to FLOAT
SQL> INSERT INTO A_TABLE VALUE (7456123.89);
1 row inserted
SQL> SELECT A FROM A_TABLE;
                       A
  7.456123890000000E+006
1 row selected
SQL> ROLLBACK;
SQL>
SQL> CREATE TABLE A_TABLE (A NUMBER(9));
%SQL-I-NO_NUMBER, A is being converted from NUMBER to INTEGER
SQL> INSERT INTO A_TABLE VALUE (7456123.89);
1 row inserted
SQL> SELECT A FROM A_TABLE;
           A
     7456124
1 row selected
SQL> ROLLBACK;
SQL>
SQL> CREATE TABLE A_TABLE (A NUMBER(9,2));
%SQL-I-NO_NUMBER, A is being converted from NUMBER to INTEGER
SQL> INSERT INTO A_TABLE VALUE (7456123.89);
1 row inserted
SQL> SELECT A FROM A_TABLE;
              A
     7456123.89
1 row selected
SQL> ROLLBACK;
SQL>
SQL> CREATE TABLE A_TABLE (A NUMBER(9,1));
%SQL-I-NO_NUMBER, A is being converted from NUMBER to INTEGER
SQL> INSERT INTO A_TABLE VALUE (7456123.89);
1 row inserted
SQL> SELECT A FROM A_TABLE;
             A
     7456123.9
1 row selected
```

```
SQL> ROLLBACK;
SQL>
SQL> CREATE TABLE A_TABLE (A NUMBER(6));
%SQL-I-NO_NUMBER, A is being converted from NUMBER to INTEGER
SQL> INSERT INTO A_TABLE VALUE (7456123.89);
%RDB-E-VALOUTRANGE, value outside the specified precision (6) for column "A"
SQL> SELECT A FROM A_TABLE;
0 rows selected
SQL> ROLLBACK;
SQL>
SQL> CREATE TABLE A_TABLE (A NUMBER(15,1));
%SQL-I-NO_NUMBER, A is being converted from NUMBER to BIGINT
SQL> INSERT INTO A_TABLE VALUE (7456123.89);
1 row inserted
SQL> SELECT A FROM A_TABLE;
                      A
              7456123.9
1 row selected
SQL> ROLLBACK;
SQL>
SQL> CREATE TABLE A_TABLE (A NUMBER(7,-2));
%SQL-I-NO_NUMBER, A is being converted from NUMBER to INTEGER
SQL> INSERT INTO A_TABLE VALUE (7456123.89);
1 row inserted
SQL> SELECT A FROM A_TABLE;
             A
       7456100
1 row selected
SQL> ROLLBACK;
SQL>
SQL> CREATE TABLE A_TABLE (A NUMBER(*,2));
%SQL-I-NO_NUMBER, A is being converted from NUMBER to BIGINT
SQL> INSERT INTO A_TABLE VALUE (7456123.89);
1 row inserted
SQL> SELECT A FROM A_TABLE;
                       A
              7456123.89
1 row selected
SQL> ROLLBACK;
```

3.2.2.3 Other Synonyms for VARCHAR

In addition to the VARCHAR2 synonyms listed in Section 3.2.2.1, the CHARACTER VARYING and CHAR VARYING data type keywords are now supported by Oracle Rdb as synonyms for VARCHAR, in compliance with the ANSI/ISO SQL standard.

3.2.2.4 LIST OF BYTE VARYING Subtypes

Oracle Rdb supports BINARY and TEXT as subtypes for the LIST OF BYTE VARYING data type. (See the *Oracle Rdb7 SQL Reference Manual* for a complete description of the LIST OF BYTE VARYING data type.) Use these subtypes to specify the data contained within a LIST OF BYTE VARYING data type. The subtype TEXT specifies that the data type can contain any printable characters. The subtype BINARY specifies that the data type contains raw binary data represented in hexadecimal notation.

3.2.3 Value Expressions

A **value expression** is a symbol or string of symbols used to represent or calculate a single value. When you use a value expression in a statement, SQL retrieves or calculates the value associated with the expression and uses that value when executing the statement.

Value expressions are also called **scalar expressions** or **expressions**.

There are several different types of value expressions:

- A literal directly specifies a value. See the *Oracle Rdb7 SQL Reference Manual* for more information.
- A parameter represents a value in a host language program or in an SQL module. See the *Oracle Rdb7 SQL Reference Manual* for more information.
- A column name represents a value contained in table rows. See the *Oracle Rdb7 SQL Reference Manual* for details on specifying value expressions with column names.
- A column select expression used as a value expression specifies a one-value result table. See the *Oracle Rdb7 SQL Reference Manual* for more information.
- A built-in function calculates values based on input value expressions. See the *Oracle Rdb7 SQL Reference Manual* for more information.

  SQL built-in functions include functions such as CAST, CURRENT_USER, and TRIM. For a complete list of built-in functions, see the *Oracle Rdb7 SQL Reference Manual*.

- An aggregate function calculates a single value for a collection of rows in a result table. See the *Oracle Rdb7 SQL Reference Manual* for more information.

  SQL aggregate functions are:

  – AVG
  – COUNT
  – MAX
  – MIN
  – STD DEV
  – SUM
  – VARIANCE

  External functions allow you to execute subprograms written in 3GL host languages in the context of an SQL statement. See the *Oracle Rdb7 SQL Reference Manual* for more information.

- SQL functions (CONCAT, CONVERT, DECODE, and SYSDATE) have been added to the Oracle Rdb SQL interface for convergence with Oracle server SQL.
- The DBKEY or ROWID keyword represents the value of an internal pointer called a database key to a table row. The ROWID keyword is a synonym to the DBKEY keyword. See the *Oracle Rdb7 SQL Reference Manual* for more information.
- A character value expression represents a value that belongs to the CHAR, CHARACTER, CHAR VARYING, CHARACTER VARYING, VARCHAR, LONG VARCHAR, NCHAR, or NCHAR VARYING data type. You can link two character value expressions together using the concatenation operator (| |).
- You can also combine certain value expressions with arithmetic operators to form a value expression.
- A substring specifies a portion of a character value expression that you can manipulate using arithmetic operators.
- A conditional expression is a form of the value expression that allows applications to return alternative information within an expression. See the *Oracle Rdb7 SQL Reference Manual* for more information.

  Conditional expressions are:

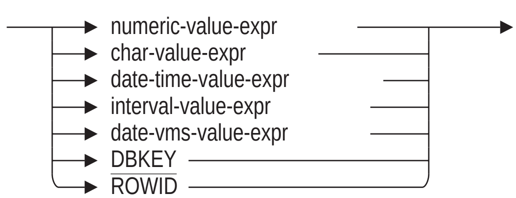
  – NULLIF

- COALESCE
- GREATEST
- LEAST
- NVL
- CASE
- DECODE

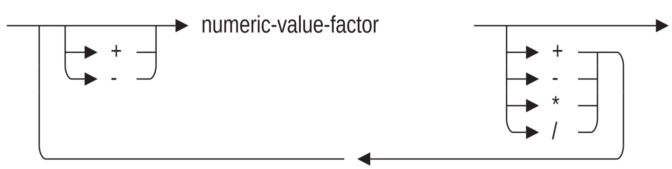
For information on the GREATEST and LEAST expressions, see Section 3.2.3.1.

The following syntax diagrams show the format of an SQL value expression:

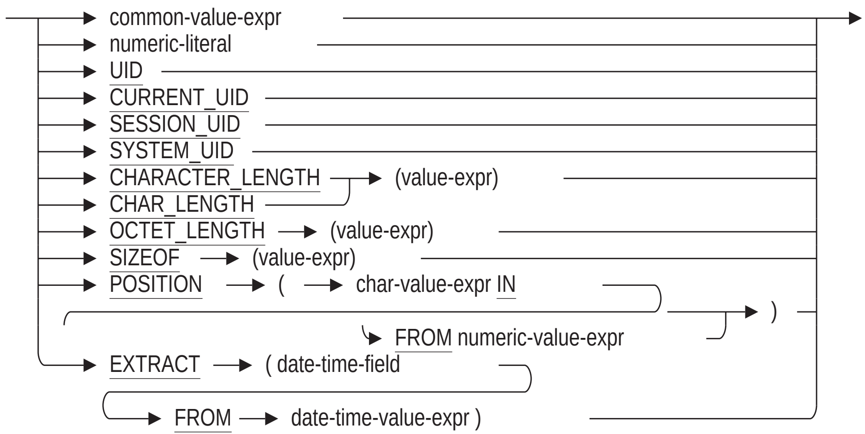
value-expr =



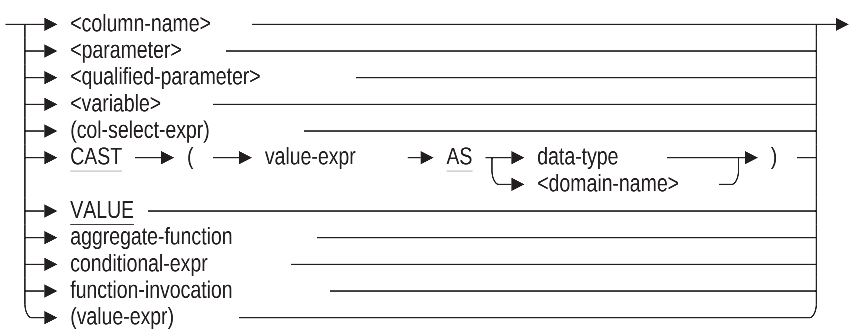
numeric-value-expr =



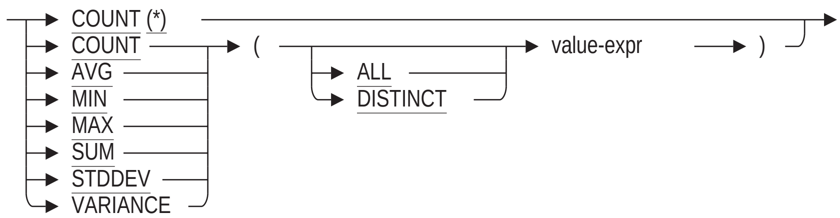
numeric-value-factor =



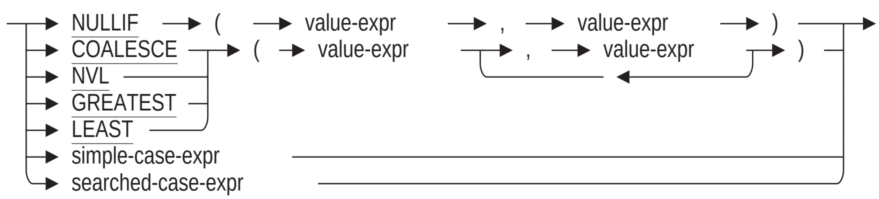
common-value-expr =



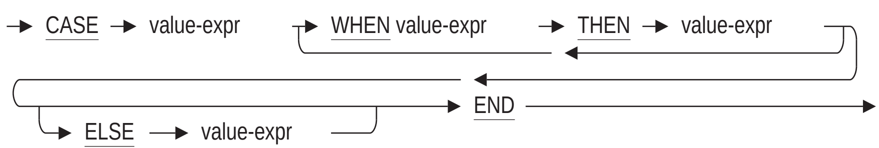
aggregate-function =



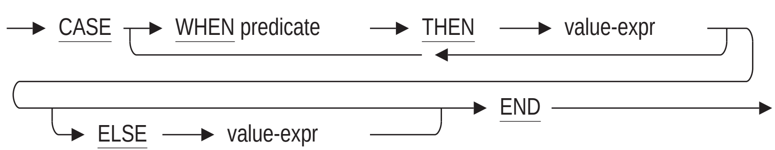
conditional-expr =



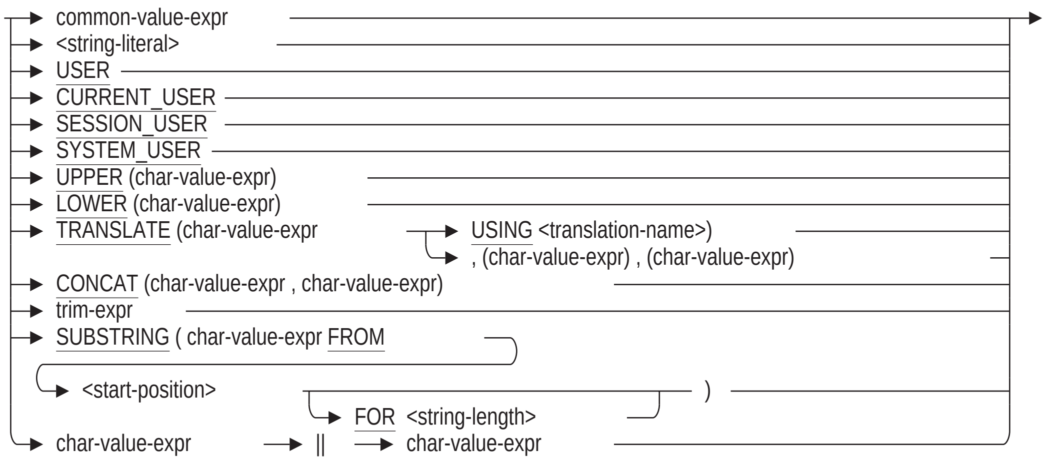
simple-case-expr =



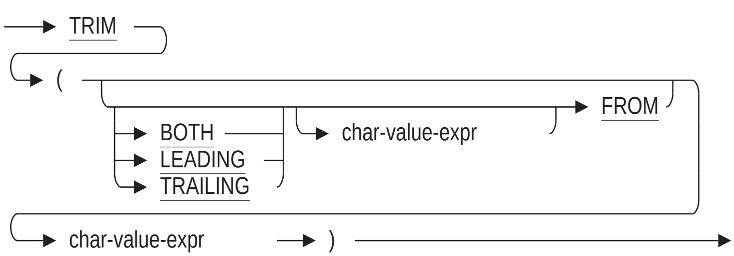
searched-case-expr =



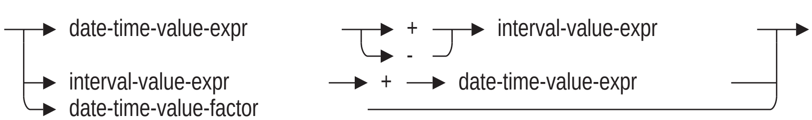
char-value-expr =



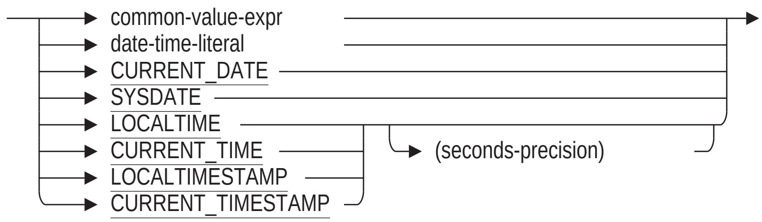
trim-expr =



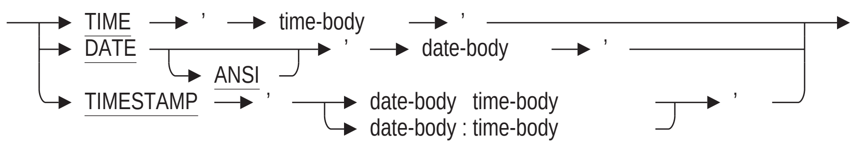
date-time-value-expr =



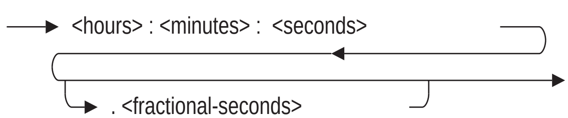
date-time-value-factor =



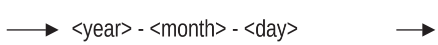
date-time-literal =



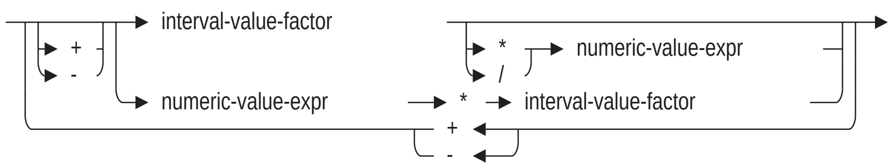
time-body =



date-body =



interval-value-expr =



interval-value-factor =

date-vms-value-expr =

For information regarding date-time data types, see the *Oracle Rdb7 SQL Reference Manual.*

3.2.3.1 GREATEST and LEAST Functions

The GREATEST and LEAST functions accept a list of two or more value expressions (all of which must be of comparable types) and return the greatest value from the list for the GREATEST function and the least value from the list for the LEAST function. The value expressions specified can be column references, subselects, function calls, literal values, and other complex value expressions.

The data type of the resulting expression is a common data type to which all value expressions in the list can be converted. For example, LEAST(10, 10.3, 123E100) results in a DOUBLE PRECISION result because at least one literal is DOUBLE PRECISION.

If the result data type resolves to a fixed CHARACTER string, then GREATEST and LEAST return a CHARACTER VARYING (also known as VARCHAR) string with the maximum length.

The NULL keyword can appear in the list but is ignored. However, not all value expressions can be specified as NULL. That is, a non-NULL value expression must be in the list so that the data type for the expression can be determined.

The GREATEST and LEAST functions can result in NULL only if at run time all value expressions result in NULL.

If DATE VMS literals are used as an argument to the GREATEST or LEAST function, the function must be prefixed with the type DATE VMS so that SQL will accept it as a DATE VMS literal and not a string literal. See Example 1.

Example 1 Using CHARACTER Versus DATE VMS Literals

```
SQL> -- Different results are returned by the LEAST function (and
SQL> -- the GREATEST function) if the parameters are treated as
SQL> -- CHARACTER or DATE VMS literals.  This is because it is
SQL> -- the data types of the source expressions that determine the
SQL> -- the result data type of the LEAST (and GREATEST) functions.
SQL> select LEAST ('1-APR-2001', '10-JAN-2000'),
cont> LEAST (DATE VMS'1-APR-2001', DATE VMS'10-JAN-2000')
cont> from rdb$database;

 1-APR-2001    10-JAN-2000 00:00:00.00
1 row selected
```

Example 2 finds the least value for the birthday of employees from two employees tables.

Example 2 Using the LEAST Function

```
SQL> SELECT LEAST (M.BIRTHDAY, E.BIRTHDAY, :SUPPLIED_DATE)
cont> FROM EMPLOYEES E, JOB_HISTORY JH, DEPARTMENTS D, EMPLOYEES M
cont> WHERE E.EMPLOYEE_ID = JH.EMPLOYEE_ID AND
cont> ...
```

3.2.4 Predicates

A **predicate** specifies a condition that SQL evaluates as true, false, or unknown. Predicates are also called conditional expressions. You can specify several different types of predicates with different conditional operators. The different types of predicates are:

- Basic
- BETWEEN
- Complex
- CONTAINING
- EXISTS
- IN
- IS NULL
- LIKE
- Quantified
- SINGLE
- STARTING WITH
- UNIQUE

When you compare character value expressions, the character sets of those value expressions must be identical.

See the *Oracle Rdb7 SQL Reference Manual* for more information on predicates. See Section 3.2.4.1 for a description of the UNIQUE predicate.

3.2.4.1 UNIQUE Predicate

The UNIQUE predicate is used to determine if duplicate rows exist in the result table of a column select expression. Note that the UNIQUE predicate (in compliance with the generic "SQL language standard") ignores rows with a NULL column value and ensures uniqueness for the other column values. Contrast this with the SINGLE predicate, which considers a single column value of NULL as a match for any other NULL value in the same column. See the *Oracle Rdb7 SQL Reference Manual* for more information on the SINGLE predicate.

```
unique-predicate =

---> UNIQUE ---> ( ---> col-select-expr ---> ) --->
```

If any two rows in the expression are equal to one another, the UNIQUE predicate evaluates to false.

Example 1 determines those cities in which one and only one employee from the EMPLOYEES database lives.

Example 1

```
SQL> SELECT E.LAST_NAME, E.CITY FROM EMPLOYEES E
cont> WHERE UNIQUE
cont> (SELECT * FROM EMPLOYEES EMP
cont> WHERE EMP.CITY=E.CITY);
 LAST_NAME        CITY
 Harrison         Boston
 Smith            Bristol
 McElroy          Cambridge
 Kilpatrick       Marlow
 Sciacca          Munsonville
 Vormelker        Rochester
 Dement           Sanbornton
 Babbin           Sanbornville
 Keisling         Twin Mountain
 Ziemke           Winnisquam
 Johnston         Wolfeboro
11 rows selected
```

3.2.5 Select Expressions

Select expressions are the basis for the SELECT, DECLARE CURSOR, CREATE VIEW, and INSERT statements. Select expressions specify a result table to be retrieved from the database or to be stored in the database, and are derived from some combination of the tables, views, or derived tables identified in the FROM clause of the expression.

Using select expressions, you can define three types of result tables:

- Simple tables
- Joined tables
- Derived tables

A **simple table** is the form of result table with which you are familiar in versions of Oracle Rdb prior to release 6.0.

A **joined table** represents a join between two table references specified in the FROM clause.

There are two types of joined tables:

- Qualified join
- Cross join

A **derived table** is a named virtual table containing data obtained through the evaluation of the FROM clause. The derived table is named by specifying the correlation name.

See the following syntax and arguments for more information on joined and derived tables.

Environment

You can use select expressions, by themselves or as part of other SQL statements, in interactive SQL or in host language programs.

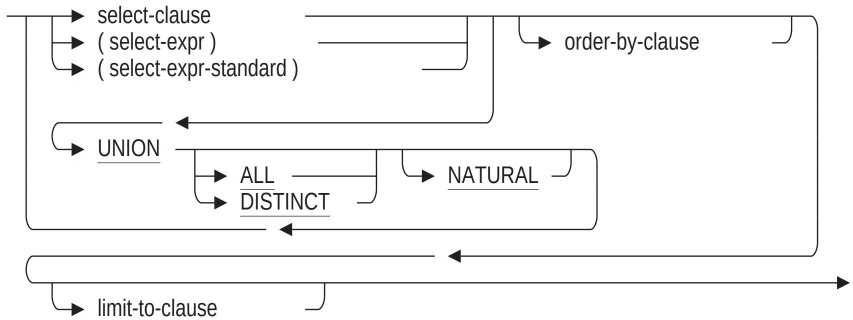
SQL evaluates the arguments in a select clause in the following order:

- FROM
- WHERE
- GROUP BY
- HAVING
- Select list
- ORDER BY
- LIMIT TO

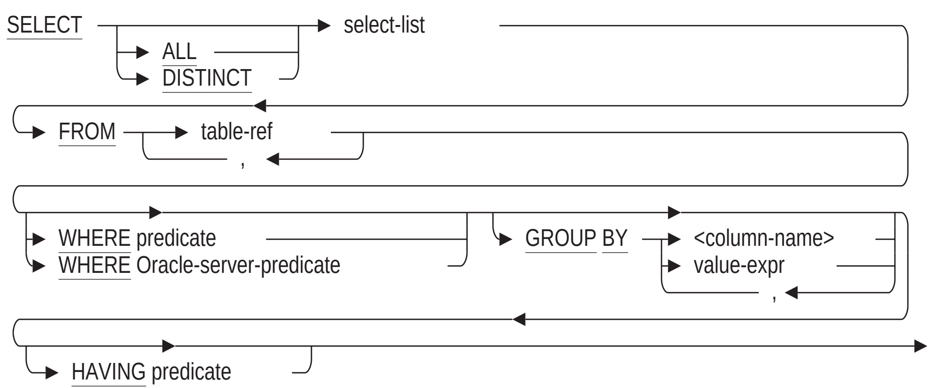
After each of these clauses, SQL produces an intermediate result table that is used in evaluating the next clause.

Format

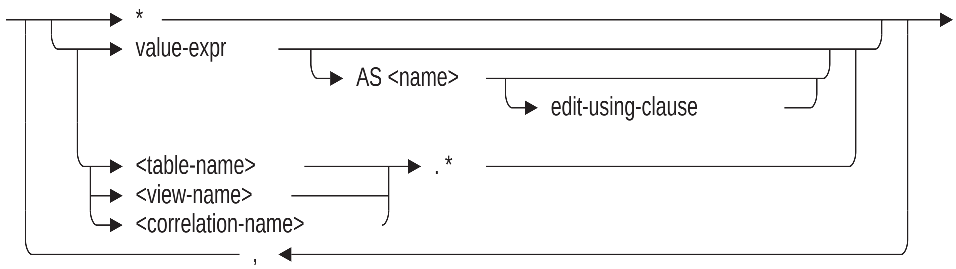
select-expr =



select-clause =

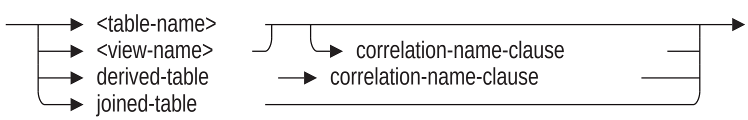


select-list =



edit-using-clause =

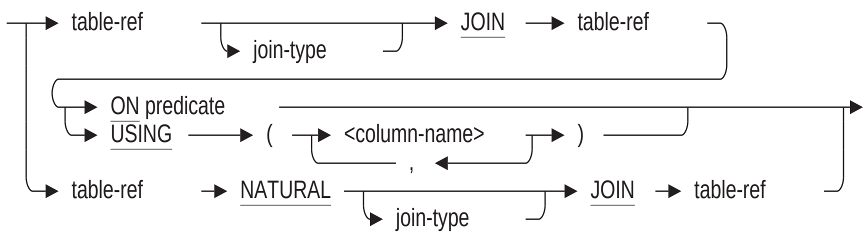
table-ref =



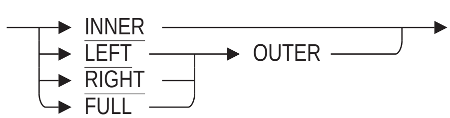
derived-table =

joined-table =

qualified-join =

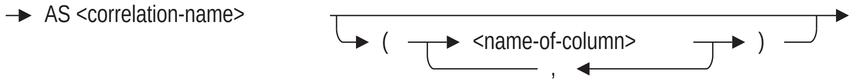


join-type =

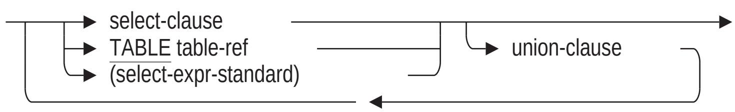


cross-join =

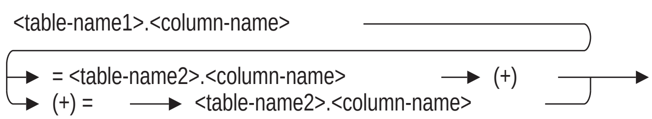
correlation-name-clause =



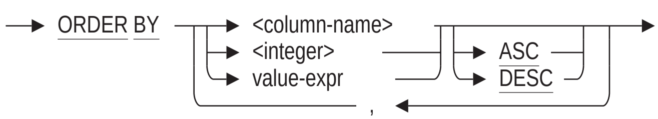
select-expr-standard =



Oracle-server-predicate =



order-by-clause =



limit-to-clause =

Arguments

EDIT USING edit-string
EDIT USING domain-name
Associates an edit string or domain name with a value expression. This clause overrides any EDIT STRING defined for the columns or variables in the query. This clause is valid in interactive SQL only. The edit string formatting characters are described in the *Oracle Rdb7 SQL Reference Manual*.

GROUP BY column-name
GROUP BY value-expr
Indicates a column name or value expression that SQL uses for organizing the intermediate result table from the WHERE clause, if specified, or the FROM clause. These groups of rows containing the same value are also called **control breaks**.

You can refer to the columns in the GROUP BY clause in two ways:

- By column name
- By a value expression that evaluates to a column name

For the first column name specified in the GROUP BY clause, SQL rearranges the rows of the preceding intermediate result table into groups whose rows all have the same value for the specified column. If a second column is specified in the GROUP BY clause, SQL then groups rows within each main group by values of the second column. If a third column is specified in the GROUP BY clause, SQL then groups rows within the secondary groups according to values of the third column. SQL groups any additional columns in the GROUP BY clause in a similar manner.

All null values in the GROUP BY clause are grouped together.

Each group is treated as the source for the values of a single row of the result table.

Because all rows of a group have the same value for the value expression specified in the GROUP BY clause, references within value expressions or predicates to that column specify a single value.

ORDER BY column-name
ORDER BY integer
ORDER BY value-expr
Specifies the order of rows for the result table. SQL sorts the rows from the intermediate result table by the values of columns or the value expression specified in the ORDER BY clause. The **intermediate result table** is the result table that SQL produces when it evaluates the preceding clause in the select expression (HAVING, GROUP BY, WHERE, or FROM).

You can refer to columns in the ORDER BY clause in two ways:

- By column name
- By column number, where the integer that you specify indicates the left-to-right position of the column in the result table

You must use an integer to identify a column in the ORDER BY clause if that column in the select list is derived from a function or an arithmetic expression.

Whether you identify columns in an ORDER BY clause using a name, using a number, or using a value expression, the columns are called **sort keys.**

When you use multiple sort keys, SQL treats the first column as the major sort key and successive columns as minor sort keys. That is, it first sorts the rows into groups based on the first value expression. Then, it uses the second value expression to sort the rows within each group, and so on. Unless you specify a sort key for every column in the result table, rows with identical values for the last sort key specified will be in arbitrary order.

Oracle-server-predicate
Allows you to form outer joins using Oracle server style syntax. The plus sign (+) operator indicates that the column preceding it is the outer join column in a join.

Usage Notes

- The following restrictions apply when you use a value expression in a GROUP BY clause:
  - You must have a syntactically similar value expression in the select list.
  - The asterisk (*) is not supported in value expressions with GROUP BY.
  - Value expressions in a GROUP BY clause are not supported in subqueries.
- The following notes apply when you use the Oracle server predicate:
  - If tables A and B are joined by multiple join conditions, then the plus (+) operator must be used in all these conditions.
  - The plus operator can be applied only to a column, not to an arbitrary expression. However, an arbitrary expression can contain a column marked with the plus operator.
  - A condition containing the plus operator cannot be combined with another condition using the OR logical operator.

- A condition cannot use the IN comparison operator to compare a column marked with the plus operator to another expression.
- A condition cannot compare a column marked with the plus operator to a subquery.
- If the WHERE clause contains a condition that compares a column from table B to a constant, then the plus operator must be applied to the column such that the rows from table A for which Oracle Rdb has generated NULLs for this column are returned.
- In a query that performs outer joins of more than two pairs of tables, a single table can only be the null-generated table for one other table. For this reason, you cannot apply the plus operator to the column of table B in the join condition for tables A and B and the join condition for tables B and C.

Examples

Example 1 Associating an Edit String with a Value Expression

```
SQL> CREATE DOMAIN MONEY INTEGER(2)
cont> EDIT STRING '$$$,$$$,$$9.99';
SQL> --Calculate the average salary for all current jobs.
SQL> SELECT EMPLOYEE_ID,
cont> AVG(SALARY_AMOUNT) AS AVERAGE EDIT USING MONEY,
cont> MAX(SALARY_AMOUNT) AS MAXIMUM EDIT USING MONEY,
cont> MAX(SALARY_START) AS START_DATE EDIT USING 'YYYBDDBMMMBWWW'
cont> FROM SALARY_HISTORY
cont> WHERE SALARY_END IS NULL
cont> GROUP BY EMPLOYEE_ID;
 EMPLOYEE_ID        AVERAGE         MAXIMUM   START_DATE
 00164           $51,712.00      $51,712.00   983 14 Jan Fri
 00165           $11,676.00      $11,676.00   982  1 Jul Thu
 00166           $18,497.00      $18,497.00   982  7 Aug Sat
 00167           $17,510.00      $17,510.00   982 21 Aug Sat
.
.
.
 00435           $84,147.00      $84,147.00   982 12 Mar Fri
 00471           $52,000.00      $52,000.00   982 15 Aug Sun
100 rows selected
```

Example 2 Using the ORDER BY Clause with a Value Expression

```
SQL> SELECT * FROM EMPLOYEES
```

```
cont> ORDER BY EXTRACT (YEAR FROM BIRTHDAY),
cont>  TRIM(FIRST_NAME) || TRIM(LAST_NAME);
 00190         O'Sullivan       Rick         G.
   78 Mason Rd.                NULL                           Fremont
      NH      03044         M      12-Jan-1923   1             None
 00231         Clairmont        Rick         NULL
   92 Madiso7 Drive            NULL                           Chocorua
      NH      03817         M      23-Dec-1924   2             None
 00183         Nash             Walter       V.
   197 Lantern Lane            NULL                           Fremont
      NH      03044         M      19-Jan-1925   1             None
 00177         Kinmonth         Louis        NULL
   76 Maple St.                NULL                           Etna
      NH      03750         M       7-Apr-1926   1             None
 00240         Johnson          Bill         R.
   20 South St                 NULL                           Milford
      NH      03055         M      13-Apr-1927   2             None
.
.
.
```

Example 3 Using the GROUP BY Clause with a Value Expression

```
SQL> SELECT COUNT (*), EXTRACT (YEAR FROM BIRTHDAY)
cont> FROM EMPLOYEES
cont> GROUP BY EXTRACT (YEAR FROM BIRTHDAY);
           1          1923
           1          1924
           1          1925
           1          1926
           4          1927
           2          1928
           1          1930
           2          1931
.
.
.
```

Example 4 Performing an Outer Join with Oracle Server Style Syntax

```
SQL> SELECT EMPLOYEES.EMPLOYEE_ID, JOB_CODE
cont> FROM EMPLOYEES, CURRENT_JOB
cont> WHERE EMPLOYEES.EMPLOYEE_ID= CURRENT_JOB.EMPLOYEE_ID(+);
 EMPLOYEES.EMPLOYEE_ID   CURRENT_JOB.JOB_CODE
```

```
 00164                 DMGR
 00165                 ASCK
 00166                 DMGR
 00167                 APGM
 00168                 DMGR
 00169                 SPGM
 00170                 SCTR
 00171                 PRGM
.
.
.
```

3.3 SQL Module Language

This section describes changes to the SQL module language.

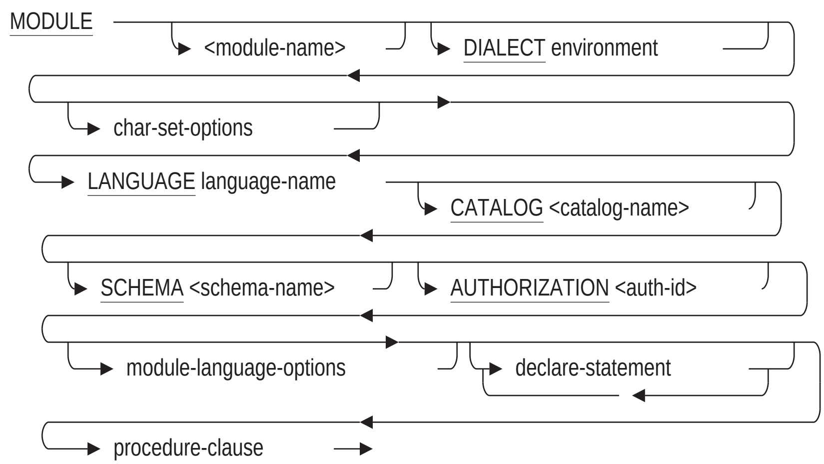
3.3.1 SQL Module Language Syntax

The SQL module language provides special keywords and syntax allowing procedures containing SQL statements to be called from host languages that are not supported by the SQL precompiler.

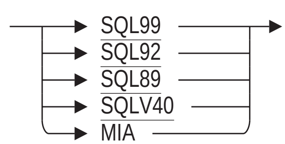
Environment

SQL module language elements must be part of an SQL module file.

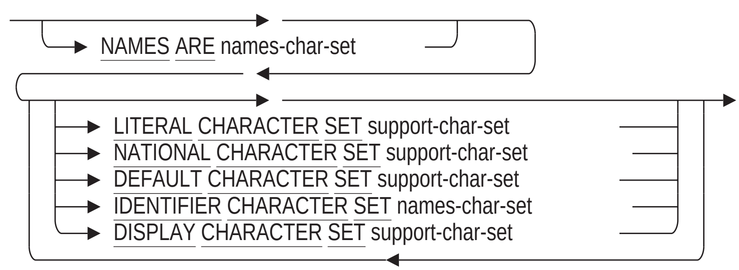
Format



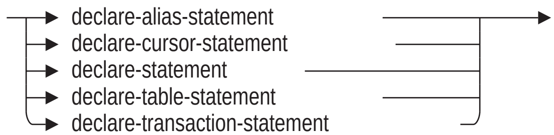
environment =



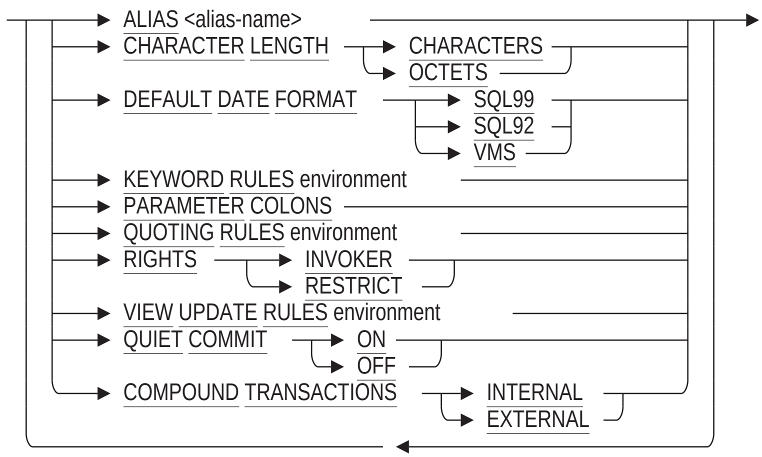
char-set-options =



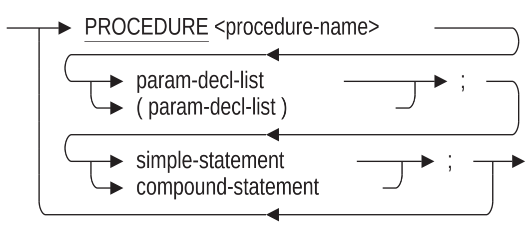
declare-statement =



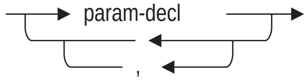
module-language-options =



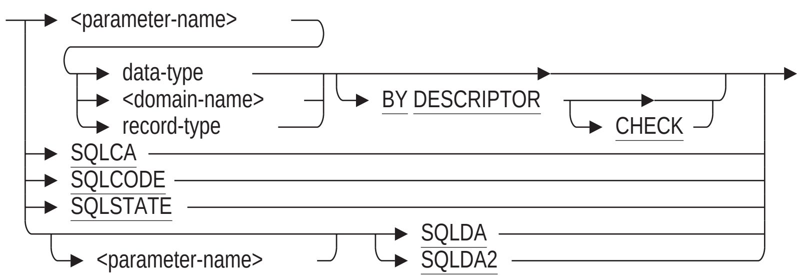
procedure-clause =



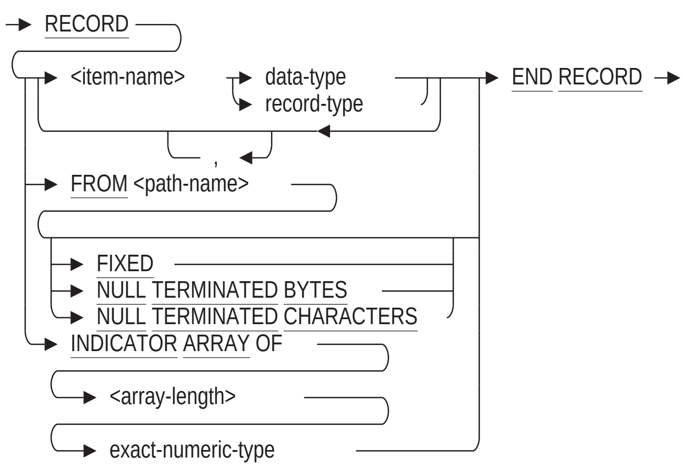
param-decl-list =



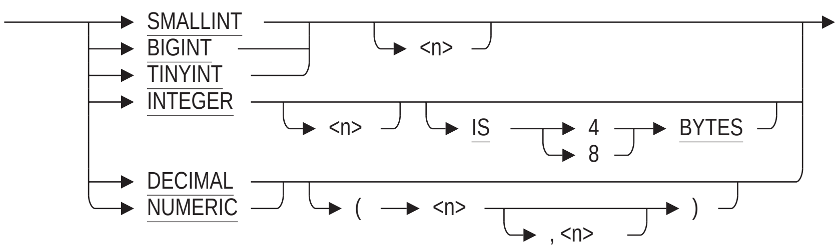
param-decl =



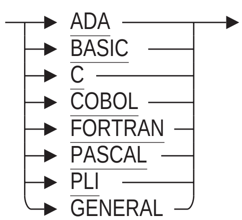
record-type =



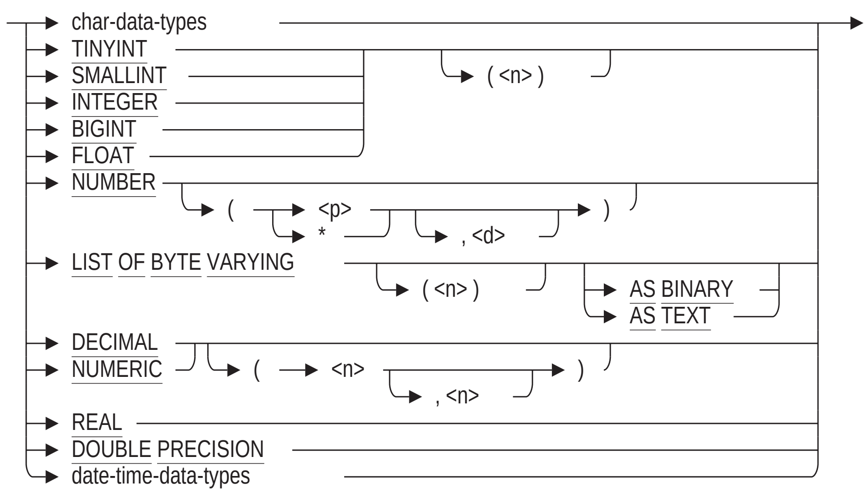
exact-numeric-type =



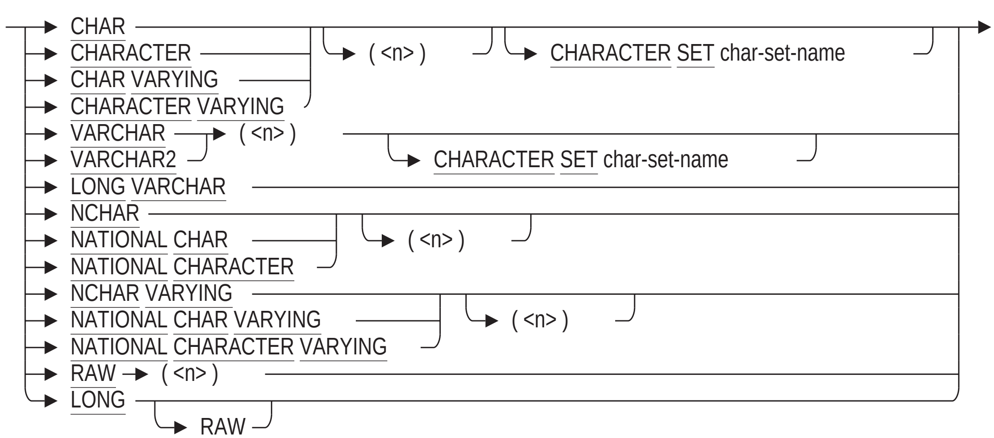
language-name =



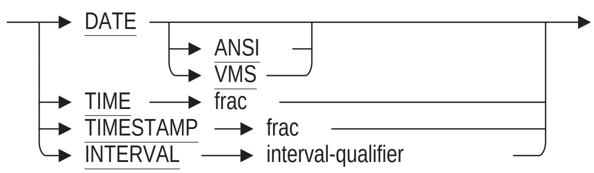
data-type =



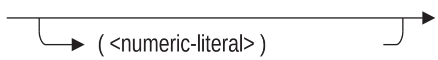
char-data-types =



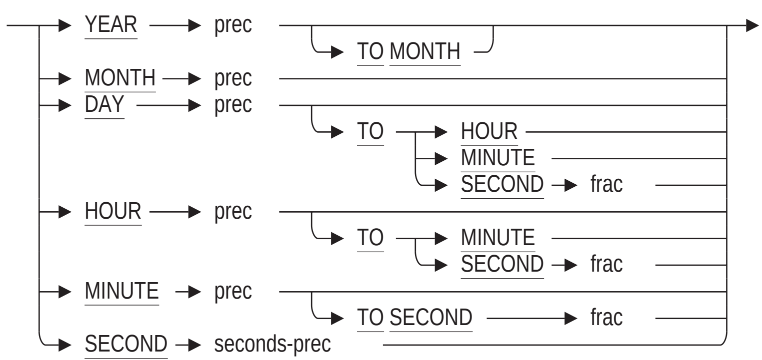
date-time-data-types =



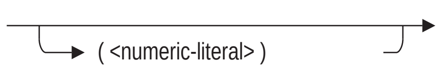
frac =



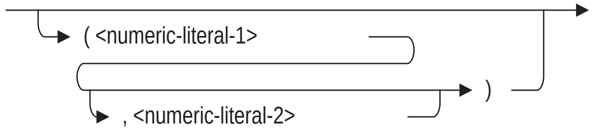
interval-qualifier =



prec =



seconds-prec =



Arguments

QUIET COMMIT ON
QUIET COMMIT OFF

The QUIET COMMIT ON clause disables error reporting for the COMMIT and ROLLBACK statements if either statement is executed when no transaction is active. The QUIET COMMIT OFF clause enables error reporting for the COMMIT and ROLLBACK statements if either statement is executed when no transaction is active. The QUIET COMMIT OFF clause is the default.

COMPOUND TRANSACTIONS INTERNAL
COMPOUND TRANSACTIONS EXTERNAL

Allows you to specify whether SQL should start a transaction before executing a compound statement or stored procedure.

The COMPOUND TRANSACTIONS EXTERNAL clause instructs SQL to start a transaction before executing a procedure. The COMPOUND TRANSACTIONS INTERNAL clause instructs SQL to allow a procedure to start a transaction as required by the procedure execution.

By default, SQL starts a transaction before executing a compound statement if there is no current transaction.

Usage Notes

No new usage notes.

Examples

Example 1 Enabling the QUIET COMMIT Option

```
MODULE txn_control
LANGUAGE BASIC
```

```
PARAMETER COLONS
QUIET COMMIT ON

PROCEDURE C_TXN (SQLCODE);
COMMIT;
```

3.4 SQL Module Language Processor Command Line

You can define a symbol to make invoking the SQL module processor easier. For example:

```
$ SQLMOD == "$SQL$MOD"
```

You then can invoke the SQL module processor with or without a module file specification:

- If you invoke the SQL module processor without a module file specification, the module processor prompts you for it. For example:

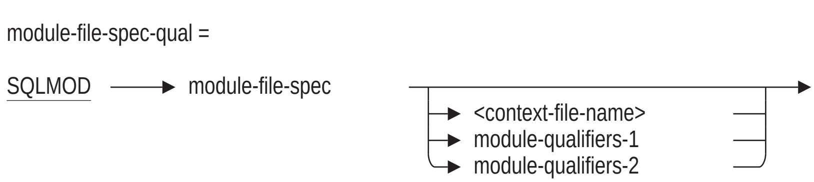
  ```
  $ SQLMOD
  INPUT FILE> module-file-specification
  ```

- If you invoke the SQL module processor with a module file specification as part of the DCL command line, SQL starts processing your module file immediately after you press the Return key. For example:

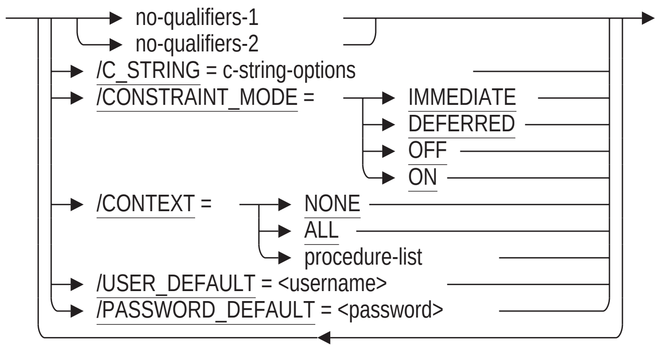
  ```
  $ SQLMOD module-file-specification
  ```

Either way, there are several qualifiers you can specify with the file specification to control how SQL processes the module file. The syntax diagram shows the format for those qualifiers.

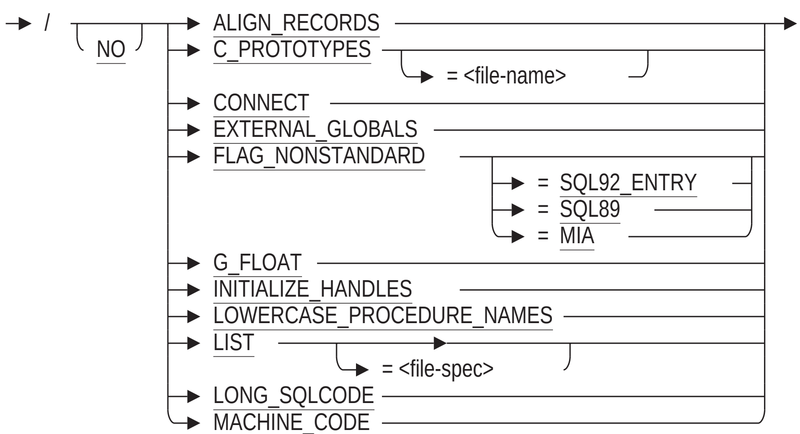
Format



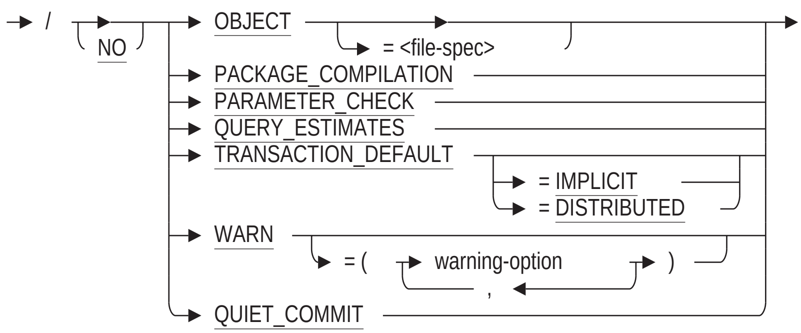
module-qualifiers-1 =



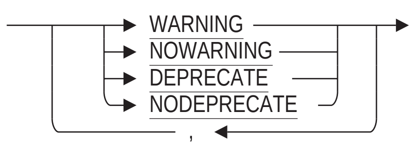
no-qualifiers-1 =



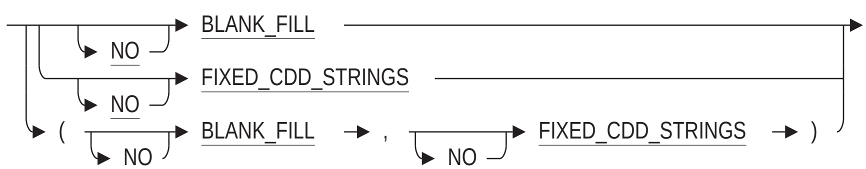
no-qualifiers-2 =



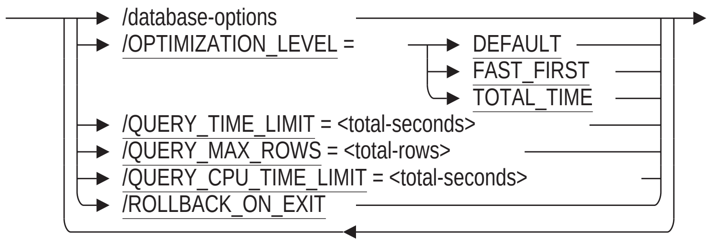
warning-option =



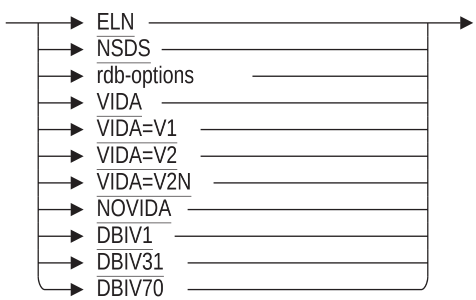
c-string-options =



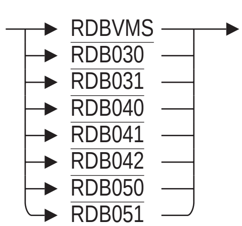
module-qualifiers-2 =



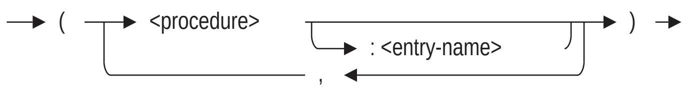
database-options =



rdb-options =



procedure-list =



Arguments

QUIET_COMMIT
NOQUIET_COMMIT
Specifies the behavior of the COMMIT and ROLLBACK statements in cases where no transaction is active.

The NOQUIET_COMMIT qualifier specifies that if there is no active transaction, SQL is to raise an error when a COMMIT or ROLLBACK statement is executed. This is the default. This default is retained for backward compatibility for applications that want to detect the situation. If the QUIET_COMMIT qualifier is specified, then a COMMIT or ROLLBACK statement executes successfully when there is no active transaction.

Usage Notes

No new usage notes.

Examples

Example 1 Using the NOQUIET_COMMIT Qualifier

```
$SQLMOD :==$SQLMOD
$SQLMOD NOTRXN.SQLMOD /NOQUIET_COMMIT
```

3.5 SQL Precompiler Command Line

You can define a symbol to help you invoke the SQL precompiler:

```
$ SQLPRE == "$SQL$PRE"
```

Because the SQL precompiler requires a language qualifier, you might want to define a particular language so that you can invoke the command on one line:

```
$ SADA == "$SQL$PRE/ADA"
```

```
$ SADA SQL_DYNAMIC
```

By defining symbols, you can invoke the SQL precompiler with or without a file specification for a host language program file:

- If you invoke the SQL precompiler without an input file specification for a host language program file, the precompiler prompts you for it. For example:

  ```
  $ SQLPRE
  INPUT FILE> pre-host-file-spec
  ```

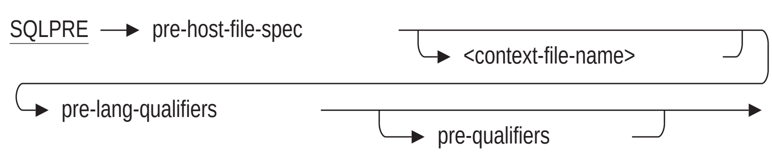
- If you invoke the SQL precompiler with a host language program file as part of the DCL command line, SQL starts processing your file immediately after you press the Return key. For example:

  ```
  $ SADA pre-host-file-spec pre-qualifiers
  ```

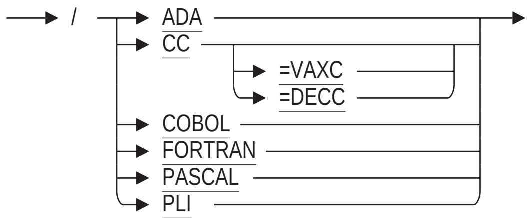
Whichever method you choose to invoke the precompiler, you have the option to specify a wide range of qualifiers that control how the SQL precompiler processes the module file. The syntax diagrams show the format for the qualifiers that you can include with the host language program file specification.

Format

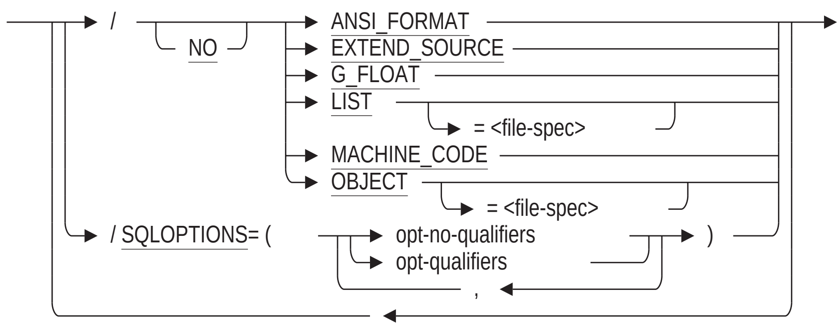
pre-host-file-qual =



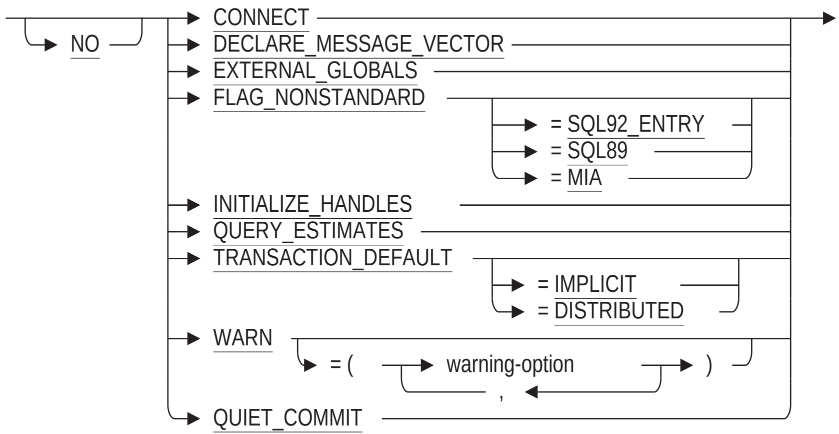
pre-lang-qualifiers =



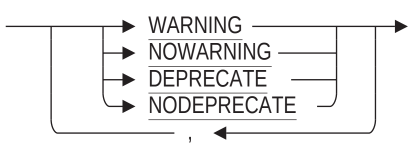
pre-qualifiers =



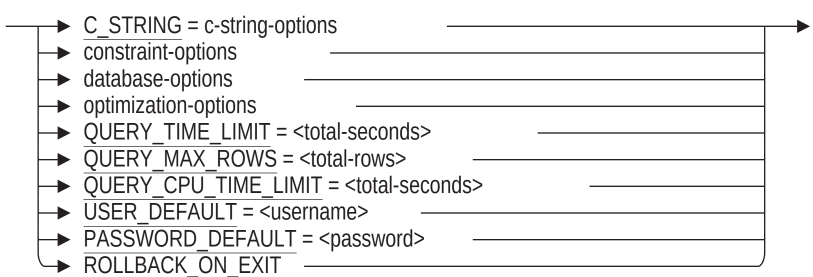
opt-no-qualifiers =



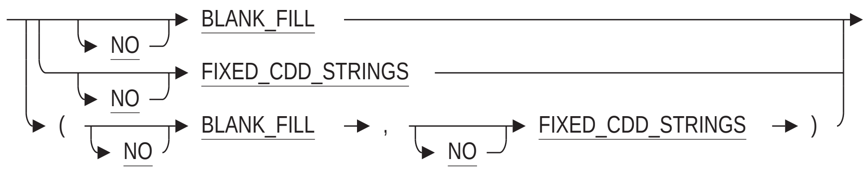
warning-option =



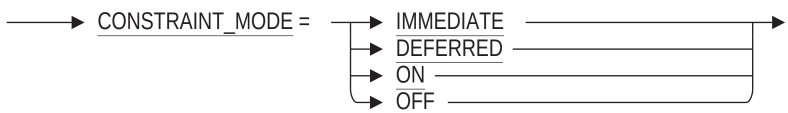
opt-qualifiers =



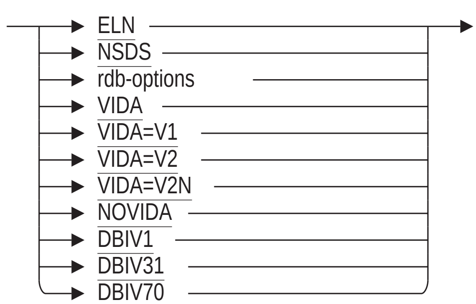
c-string-options =



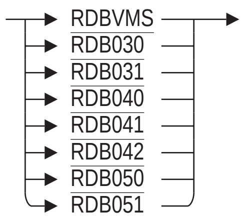
constraint-options =



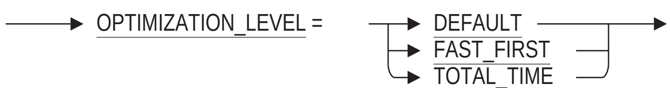
database-options =



rdb-options =



optimization-options =



Arguments

/SQLOPTIONS=QUIET_COMMIT
/SQLOPTIONS=NOQUIET_COMMIT
Specifies the behavior of the COMMIT and ROLLBACK statements in cases where no transaction is active.

By default, if there is no active transaction, SQL raises an error when COMMIT or ROLLBACK is executed. This default is retained for backward compatibility for applications that want to detect the situation. If the SQLOPTIONS=QUIET_COMMIT qualifier is specified, then a COMMIT or ROLLBACK statement executes successfully when there is no active transaction. If the SQLOPTIONS=NOQUIET_COMMIT qualifier is specified, then a COMMIT or ROLLBACK statement returns an error if there is no active transaction.

3.5.1 SQL Precompiler Supported Variable Declarations

Table 3–2 lists the Compaq Fortran variable declarations supported by the SQL precompiler on OpenVMS.

Table 3–2 Compaq Fortran Declarations Supported by the SQL Precompiler

Fortran Type	SQL Type	Comments
BYTE	TINYINT	None.
CHARACTER**n*	CHAR	The *n* represents a positive integer literal.
INTEGER	INTEGER	None.
INTEGER*1	TINYINT	This is identical to the BYTE and LOGICAL*1 Fortran data types.
INTEGER*2	SMALLINT	None.
INTEGER*4	INTEGER	None.
INTEGER*8	BIGINT	None.
LOGICAL	INTEGER	None.
LOGICAL*1	TINYINT	None.
LOGICAL*2	SMALLINT	None.
LOGICAL*4	INTEGER	None.
LOGICAL*8	BIGINT	None.
REAL	REAL	None.
REAL*4	REAL	None.
REAL*8	DOUBLE PRECISION	None.
STRUCTURE /name/ integer*2 len character**n* body END STRUCTURE	VARCHAR	The len component of the structure must be set to the correct length of the string before it is used as a parameter to SQL. The *n* represents a positive integer literal.

Table 3–3 lists the Compaq C variable declarations supported by the SQL precompiler on OpenVMS.

Table 3–3 Compaq C Declarations Supported by the SQL Precompiler

Compaq C Type or typedef	SQL Type	Comments
char	CHARACTER	None.

Table 3–3 (Cont.) Compaq C Declarations Supported by the SQL Precompiler(Cont.)

Compaq C Type or typedef	SQL Type	Comments
int	INTEGER	None.
short	SMALLINT	None.
float	REAL	None.
double	DOUBLE PRECISION	None.
enum	INTEGER	None.
long	INTEGER or BIGINT	The data type long is 32 bits.
int8	TINYINT	Requires #include <ints.h>.
int16	SMALLINT	Requires #include <ints.h>.
_ _int16	SMALLINT	None.
int32	INTEGER	Requires #include <ints.h>.
_ _int32	INTEGER	None.
int64	BIGINT	Requires #include <ints.h>.
_ _int64	BIGINT	None.

3.6 SQL Statements

The following sections describe new and changed SQL statements; only the statements and clauses that are new or that have changed are included here.

If a statement or clause is included here, the complete statement or clause introduction, environment, and format information is included. However, only those arguments, usage notes, and examples that are relevant to the new feature are documented here. See the *Oracle Rdb7 SQL Reference Manual* for the arguments, usage notes, and examples for features that existed prior to release 7.1.

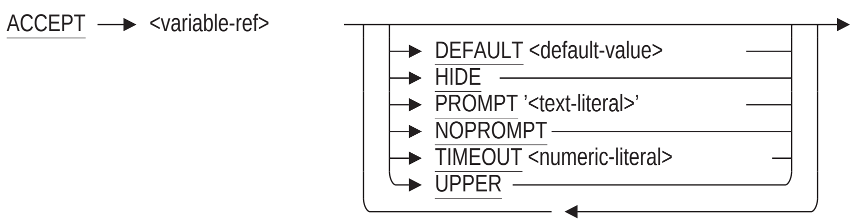
ACCEPT Statement

Prompts the user for additional information. This information is stored in an interactive SQL variable, which can subsequently be used by DML and some SET statements.

Environment

You can use the ACCEPT statement in interactive SQL.

Format



variable-ref =

Arguments

DEFAULT string-literal
Provides a default value to be used if the user presses the Return key. The default value must be a correctly formatted character string that can be converted to the data type of the variable.

HIDE
Disables echo of the input text. The default is to echo all input characters.

PROMPT string-literal
Provides a prompt string that is displayed before accepting input.

NOPROMPT
Disables prompting with a string.

TIMEOUT numeric-literal
If the user does not respond within this many seconds, then an error is returned. Negative or zero values of the numeric-literal are ignored. The default is wait indefinitely.

UPPER
All lowercase characters are converted to uppercase before assignment to the variable. The default is to leave lowercase characters unchanged.

Usage Notes

- The variable must be declared using the DECLARE syntax in interactive SQL. ACCEPT does not create this variable automatically. The leading colon (:) required for variables references is optional in the ACCEPT command.

  The variable reference may include an indicator variable that can be used to detect NULL value assignments.

- When PROMPT is specified, the character string literal (up to a maximum of 80 octets) is used to prompt the user for input.

  When NOPROMPT is specified, then the user is not given a prompt string.

  If neither PROMPT nor NOPROMPT is specified, SQL will create a prompt containing the name of the variable. For instance:

  ```
  SQL> ACCEPT :NAME;
  Enter a value for NAME: Jeff
  ```

  The prompt string does not appear in the output created by the SET OUTPUT statement. Use the PRINT command to verify the input in such cases.

- This statement is based upon the ACCEPT statement of Oracle SQL*Plus. The following SQL*Plus clauses are not currently available in Oracle Rdb SQL: NUMBER, CHAR, DATE, and FORMAT.

- If the user enters no data, but presses the Return key, then the DEFAULT value is used in place of the response. If there was no DEFAULT specified, then a zero length string value is used, which may result in a valid value for numeric variables (zero) and string variables (spaces or the empty string), but will result in errors for DATE, TIME, TIMESTAMP, and INTERVAL variables.

- If the user enters EXIT (Ctrl/Z), then either an error is raised, or if an indicator variable is provided, then the value of the variable will be set to -1 (indicating NULL). The value of the variable will be undefined.

```
SQL> ACCEPT :id INDICATOR :id_ind PROMPT 'Id? ';
Id? <exit>
SQL> PRINT :id, :id_ind;
          ID        ID_IND
           0            -1
```

- If a timeout occurs and an indicator variable is provided, then the command succeeds and the indicator is set to -1 (indicating NULL). The value of the variable will be undefined.

```
SQL> DECLARE :row_out INTEGER;
SQL> ACCEPT :row_out TIMEOUT 5;
Enter value for ROW_OUT:
%SQL-F-UNDEFVAR, Variable ROW_OUT is not defined
-SYSTEM-F-TIMEOUT, device timeout
```

- If incompatible or incorrectly formatted data is entered, then an error will be issued and the contents of the variable will be unchanged.

```
SQL> DECLARE :age INTEGER;
SQL> ACCEPT :age;
Enter value for AGE: thirty
%RDB-E-ARITH_EXCEPT, truncation of a numeric value at runtime
-COSI-F-INPCONERR, input conversion error
```

Values for DATE VMS, DATE ANSI, TIME, TIMESTAMP, and INTERVAL data types must include all required punctuation.

```
SQL> DECLARE :end_time TIME(2);
SQL> ACCEPT :end_time;
Enter value for END_TIME: 10:30
%SQL-F-DATCONERR, Data conversion error for string '10:30'
-COSI-F-IVTIME, invalid date or time
```

In this example, the required minutes and second fields of the value were omitted.

- ACCEPT reads from SYS$COMMAND, which often defaults to the same input source as SYS$INPUT. However, it is possible to have SQL read from a separate source. In this example, SQL reads the SQL statements from SUMMARY_REPORT.SQL and accepts the answers from the file ANSWERS.DAT.

```
$ define/user sys$command answers.dat
$ sql$ @summary_report
```

When the input source is not an interactive device, the PROMPT clause is ignored and the prompt does not appear in the output.

- Up to 20 previous inputs are available for command recall.

Examples

Example 1 Prompting Based on the PROMPT and NOPROMPT Clauses

```
SQL> DECLARE :x INTEGER;
SQL> DECLARE :y INTEGER;
SQL>
SQL> ACCEPT :x indicator :y PROMPT 'what value? ';
what value? 10
SQL> PRINT :x, :y;
           X             Y
          10             0
SQL>
SQL> ACCEPT :x INDICATOR :y NOPROMPT;
11
SQL> PRINT :x, :y;
           X             Y
          11             0
SQL>
SQL> ACCEPT :x;
Enter value for X: 12
SQL> PRINT :x;
           X
          12
SQL>
```

Example 2 Using ACCEPT to Prompt for SET FLAGS String

This sequence would be included in a script.

```
SQL> DECLARE :debug_flags CHAR(20);
SQL> ACCEPT :debug_flags;
Enter value for DEBUG_FLAGS: trace
SQL> PRINT :debug_flags;
 DEBUG_FLAGS
```

```
 trace
SQL> SET FLAGS :debug_flags;
SQL> SHOW FLAGS

Alias RDB$DBHANDLE:
Flags currently set for Oracle Rdb:
   PREFIX,TRACE,MAX_RECURSION(100)
```

ALTER DATABASE Statement

Alters a database in any of the following ways:

- For single-file and multifile databases, the ALTER DATABASE statement changes the characteristics of the database root file.

  The ALTER DATABASE statement lets you override certain characteristics specified in the database root file parameters of the CREATE DATABASE statement, such as whether or not a snapshot file is disabled. In addition, ALTER DATABASE lets you control other characteristics that you cannot specify in the CREATE DATABASE database root file parameters, such as whether or not after-image journaling is enabled.

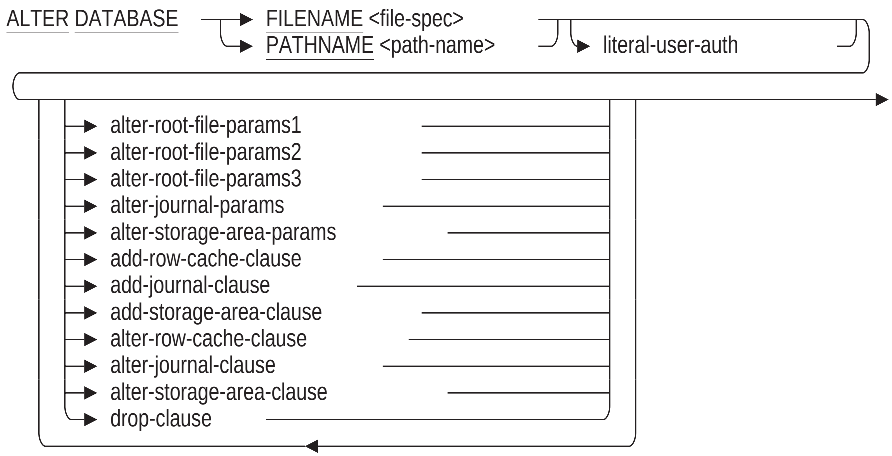
- For single-file and multifile databases, the ALTER DATABASE statement changes the storage area parameters.
- For multifile databases *only*, the ALTER DATABASE statement adds, alters, or deletes storage areas.

Environment

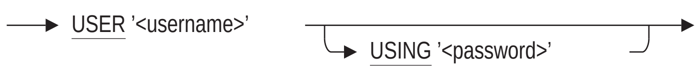
You can use the ALTER DATABASE statement:

- In interactive SQL
- Embedded in host language programs to be precompiled
- As part of a procedure in an SQL module
- In dynamic SQL as a statement to be dynamically executed

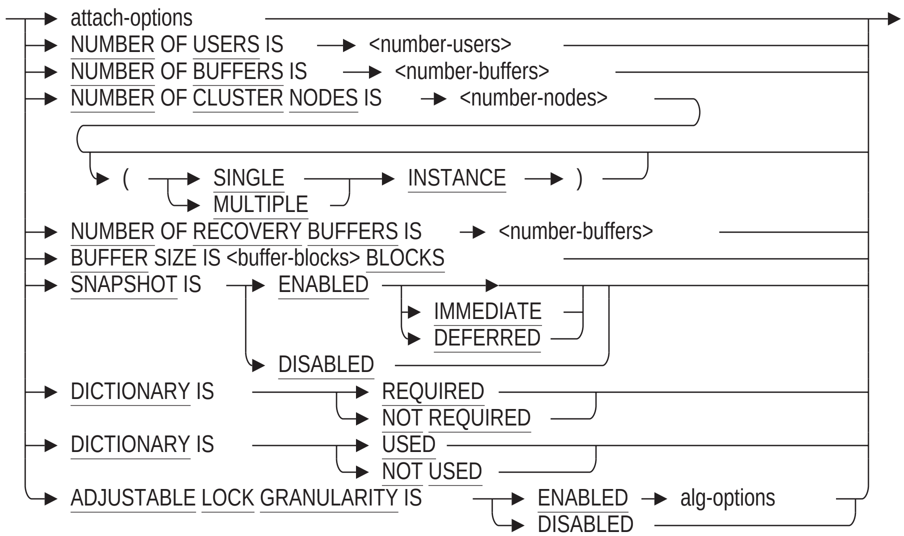
Format



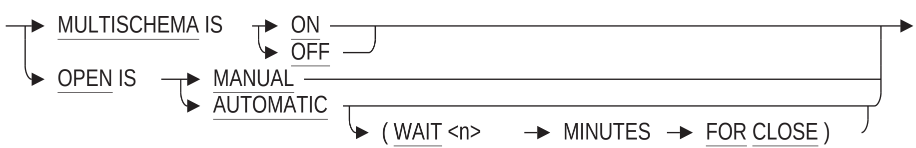
literal-user-auth =



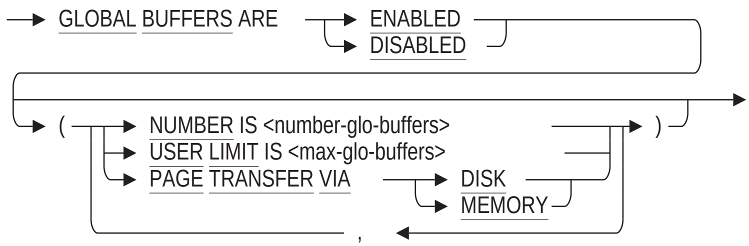
alter-root-file-params1 =



attach-options =

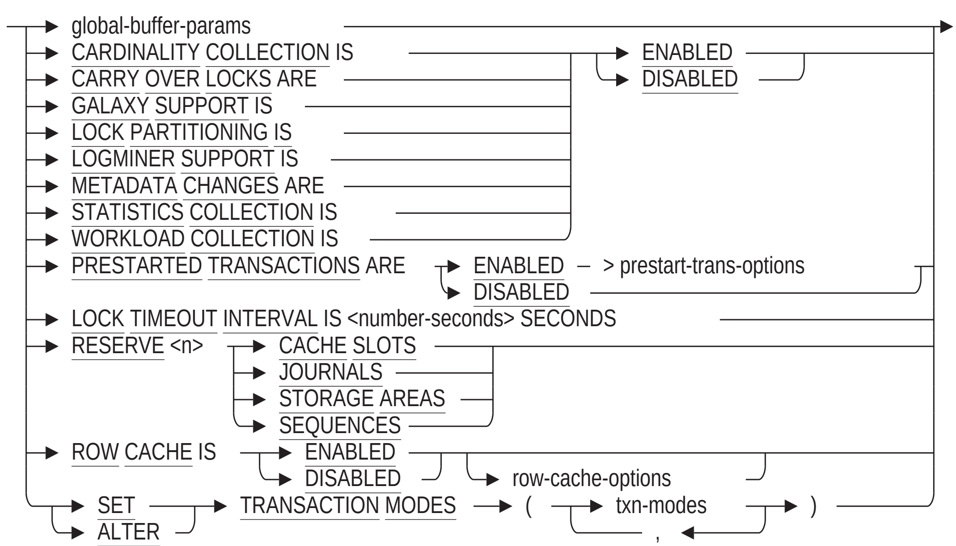


global-buffer-params=



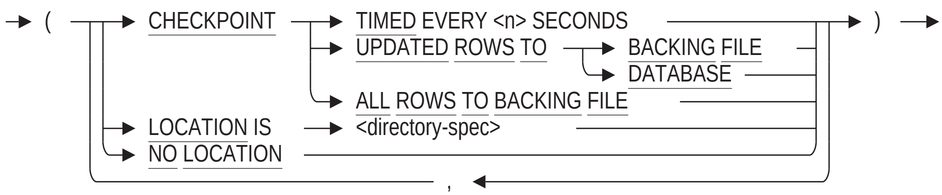
alg-options =

alter-root-file-params2 =

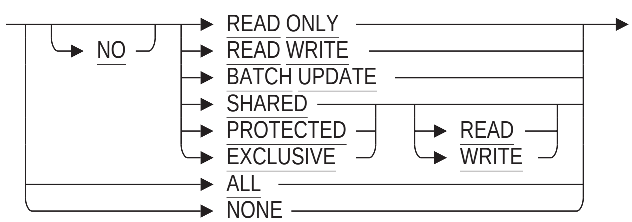


prestart-trans-options =

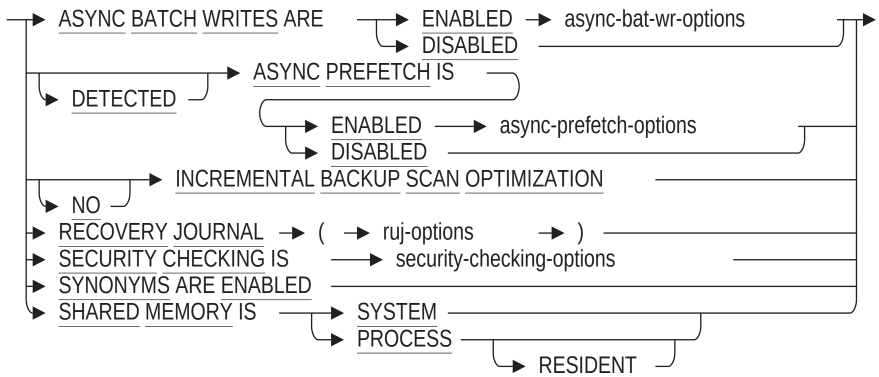
row-cache-options =



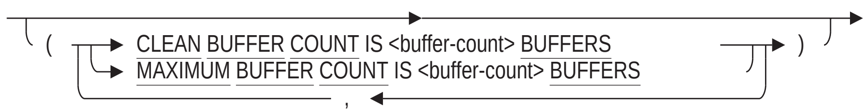
txn-modes =



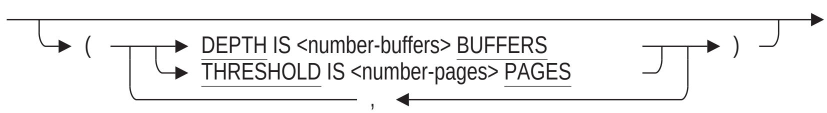
alter-root-file-params3 =



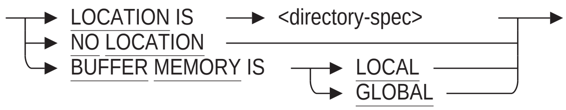
asynch-bat-wr-options =



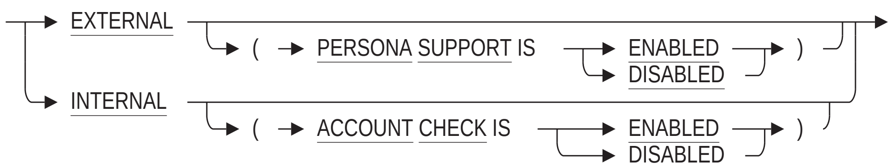
async-prefetch-options =



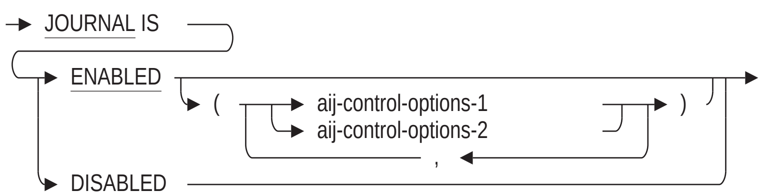
ruj-options =



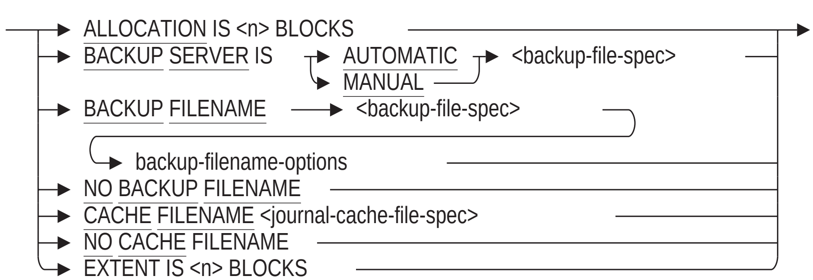
security-checking-options =



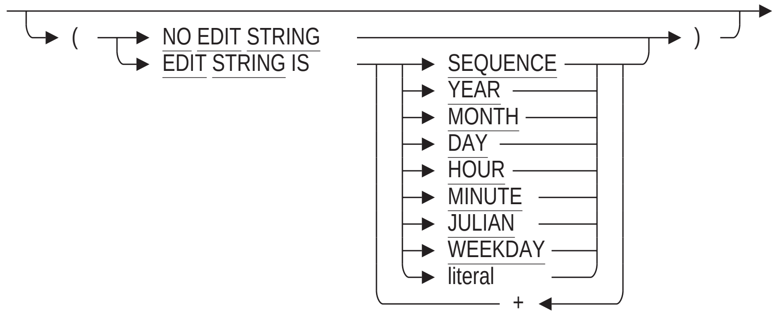
alter-journal-params =



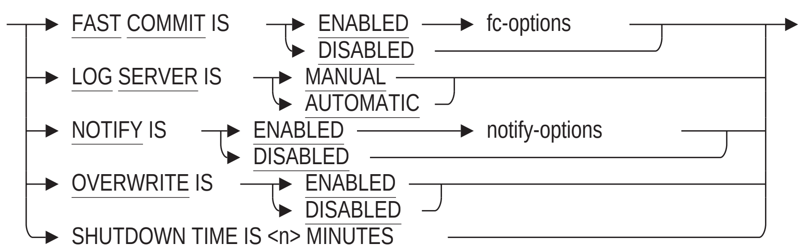
aij-control-options-1 =



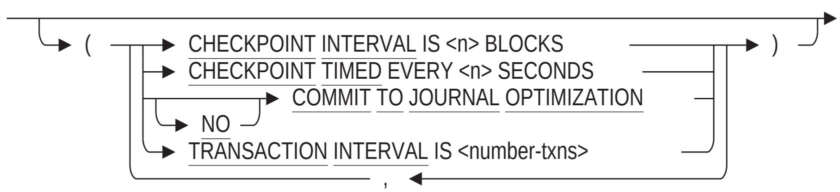
backup-filename-options =



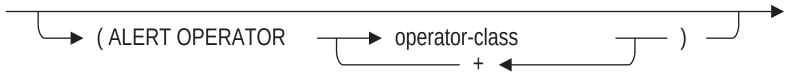
aij-control-options-2 =



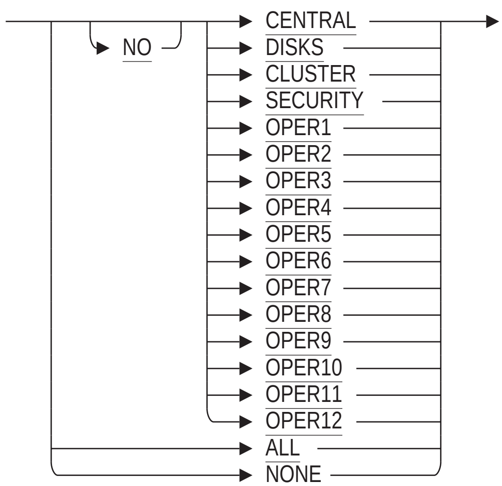
fc-options =



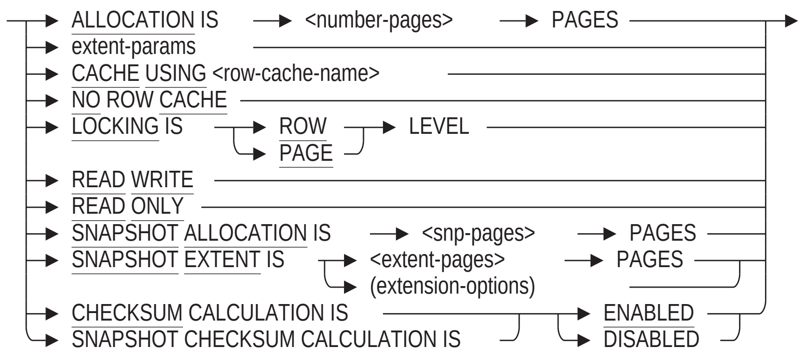
notify-options =



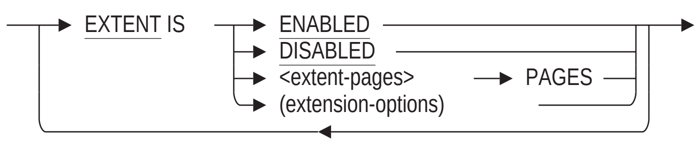
operator-class =



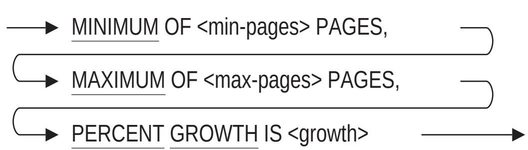
alter-storage-area-params =



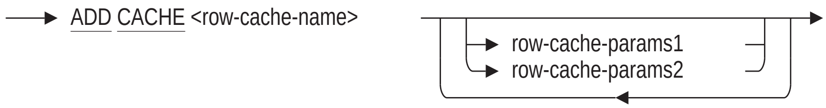
extent-params =



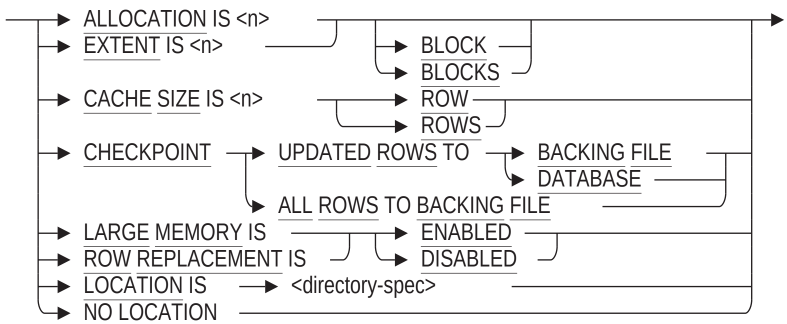
extension-options =



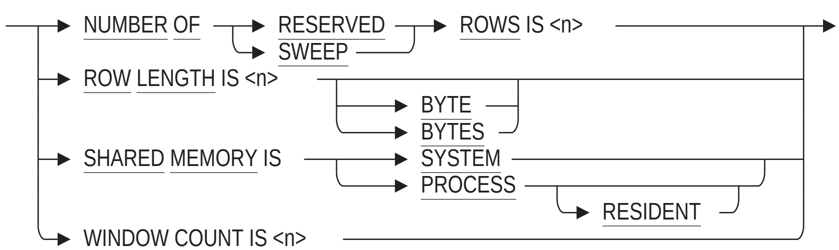
add-row-cache-clause =



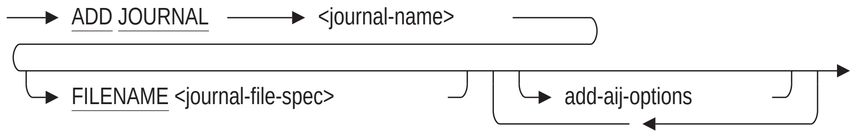
row-cache-params1 =



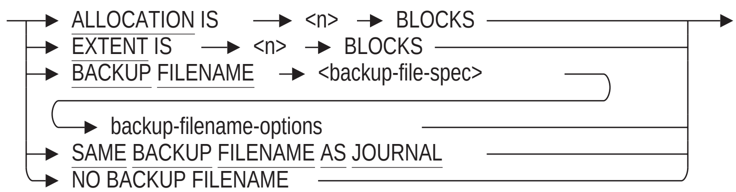
row-cache-params2 =



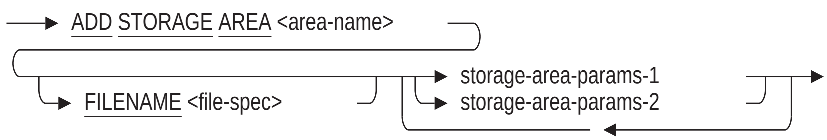
add-journal-clause =



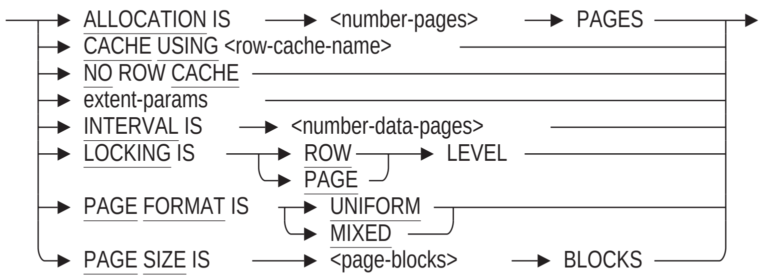
add-aij-options =



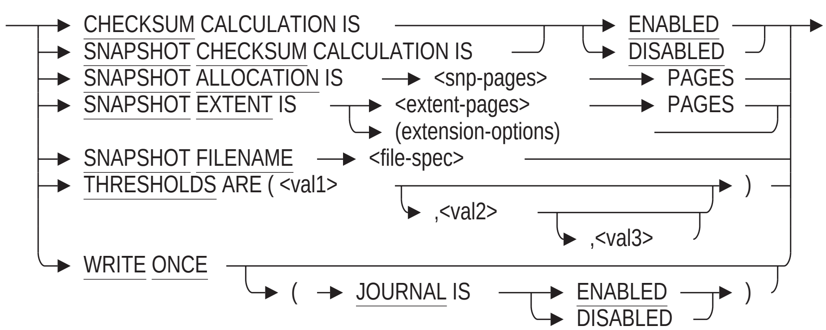
add-storage-area-clause =



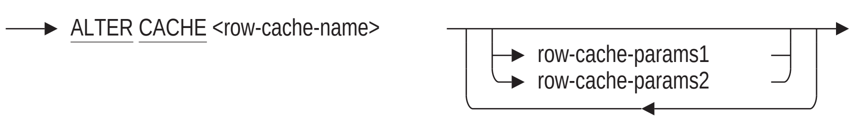
storage-area-params-1 =



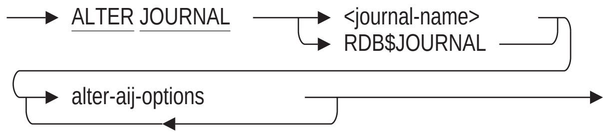
storage-area-params-2 =



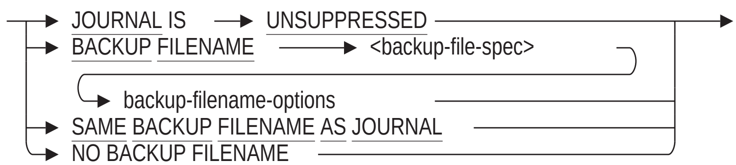
alter-row-cache-clause =



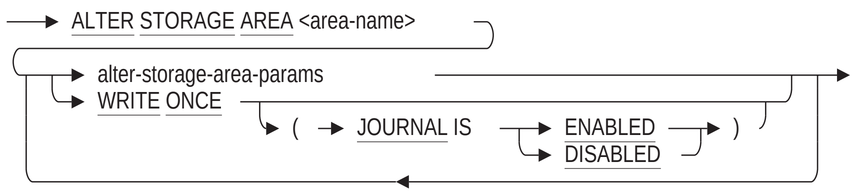
alter-journal-clause =



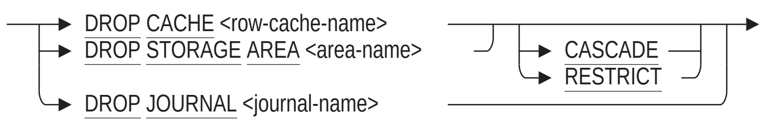
alter-aij-options =



alter-storage-area-clause =



drop-clause =



Arguments

NUMBER OF CLUSTER NODES IS n (SINGLE INSTANCE)
NUMBER OF CLUSTER NODES IS n (MULTIPLE INSTANCE)
When the Rdb root file data structures (.rdb) are mapped to shared memory, this is known as an Rdb instance. When there is only one copy of shared memory containing root file information, you can enable several optimizations to reduce locking and root file I/O during database activity. (This information may also include global buffers if they are enabled for the database.)

When NUMBER OF CLUSTER NODES is set to 1, it is implicit that a single instance is in shared memory. In an OpenVMS Galaxy configuration, because multiple nodes can access the database, you must set NUMBER OF CLUSTER NODES to a value larger than 1 to disable these single-instance optimizations.

In this environment, the database administrator may decide that only nodes that are part of this Galaxy system can access the database and, therefore, may want the optimizations normally found in single-node environments. The SINGLE INSTANCE option must be used to enable this behavior when NUMBER OF CLUSTER NODES is set to greater than 1.

MULTIPLE INSTANCE means that the Rdb root file data structures are mapped to different (or multiple) memory instances and are kept consistent through disk I/O.

MULTIPLE INSTANCE is the default.

GALAXY SUPPORT IS ENABLED
GALAXY SUPPORT IS DISABLED
Allows global memory to be shared in an OpenVMS Galaxy configuration. See *Oracle Rdb Release Notes*, release 7.1, Section 1.3.1, "Oracle Rdb Support for Compaq Galaxy Software Architecture" for more details.

LOGMINER SUPPORT IS ENABLED
LOGMINER SUPPORT IS DISABLED
Allows additional information to be written to the after-image journal file to allow the use of the RMU Unload After_Image command. See *Oracle Rdb Release Notes*, release 7.1, Section 1.5.1, "LogMiner for Rdb" for more details.

PRESTARTED TRANSACTIONS ARE ENABLED (prestart-trans-options)
Enables the prestarting of transactions.

Note that the keyword ON, available in previous versions, is synonymous with ENABLED.

With this release of Oracle Rdb, this clause is now used to establish a permanent database setting for prestarted transactions. In prior versions, this clause was only used to temporarily set the mode for prestarted transaction for the implicit attach performed by the CREATE DATABASE and IMPORT DATABASE statements.

The prestart-trans-options can be one of the following clauses:

- WAIT n SECONDS FOR TIMEOUT

  The n represents the number of seconds to wait before aborting the prestarted transaction. Timing out the prestarted transaction may prevent snapshot file growth in environments where servers stay attached to the database with long periods of inactivity.

- WAIT n MINUTES FOR TIMEOUT

  The n represents the number of minutes to wait before aborting the prestarted transaction.

- NO TIMEOUT

  This is the default for a prestarted transaction.

PRESTARTED TRANSACTIONS ARE DISABLED

Disables the prestarting of transactions.

Note that the keyword OFF, available in previous versions, is synonymous with DISABLED.

RESERVE n CACHE SLOTS

Specifies the number of row caches for which slots are reserved in the database.

You can use the RESERVE n CACHE SLOTS clause to reserve slots in the database root file for future use by the ADD CACHE clause. You can only add row caches if row cache slots are available. Slots become available after you issue a DROP CACHE clause or a RESERVE n CACHE SLOTS clause.

You cannot reduce the number of reserved slots for row caching. If you reserve 10 slots and later reserve 5 slots, a total of 15 slots are reserved for row caches.

To reserve row cache slots, you must have exclusive access to the database (it is an offline operation).

RESERVE n SEQUENCES

Specifies the number of sequences for which slots are reserved in the database. Sequences are reserved in multiples of 32. Thus, if you specify a value less than 32 for n, 32 slots are reserved. If you specify a value of 33, 64 slots are reserved, and so on.

You can use the RESERVE n SEQUENCES clause to reserve slots in the database root file for future use by the statement. Sequences can be created only if sequence slots are available. Slots become available after a DROP SEQUENCE statement or a RESERVE n SEQUENCES clause of the ALTER DATABASE statement is executed.

ROW CACHE IS ENABLED
ROW CACHE IS DISABLED

Specifies whether or not the row caching feature is enabled.

Enabling row caching does not affect database operations until a cache is created and assigned to one or more storage areas.

When the row caching is disabled, all previously created and assigned caches remain and will be available if row caching is enabled again.

To enable or disable row caching, you must have exclusive access to the database (it is an offline operation).

RECOVERY JOURNAL (BUFFER MEMORY IS LOCAL)
RECOVERY JOURNAL (BUFFER MEMORY IS GLOBAL)

Specifies whether RUJ buffers will be allocated in global or local memory.

The RUJ buffers used by each process are normally allocated in local virtual memory. With the introduction of row caching, these buffers now can be assigned to a shared global section (global memory) on OpenVMS, so that the recovery process can process this in-memory buffer and possibly avoid a disk access.

You can define this buffer memory to be global to improve row caching performance for recovery. If row caching is disabled, then buffer memory is always local.

CHECKPOINT TIMED EVERY n SECONDS

Specifies the frequency with which the row cache server (RCS) process checkpoints the contents of the row caches back to disk. The RCS process does not use the checkpoint frequency options of the FAST COMMIT clause.

The frequency of RCS checkpointing is important in determining how much of an .aij file must be read during a recovery operation following a node failure. It also affects the frequency with which marked records get flushed back to the database for those row caches that checkpoint to the database. The default is every 15 minutes (900 seconds).

CHECKPOINT UPDATED ROWS TO BACKING FILE
CHECKPOINT UPDATED ROWS TO DATABASE
CHECKPOINT ALL ROWS TO BACKING FILE

Specifies the default source and target during checkpoint operations for all row caches. If ALL ROWS is specified, then the source records written during each checkpoint operation are both the modified and the unmodified rows in a row cache. If UPDATED ROWS is specified, then just the modified rows in a row cache are checkpointed each time.

If the target of the checkpoint operation is BACKING FILE, then the RCS process writes the source row cache entries to the backing (.rdc) files. The row cache LOCATION, ALLOCATION, and EXTENT clauses are used to create the backing files. Upon recovery from a node failure, the database recovery process is able to repopulate the row caches in memory from the rows found in the backing files.

If the target is DATABASE, then updated row cache entries are written back to the database. The row cache LOCATION, ALLOCATION, and EXTENT clauses are ignored. Upon recovery from a node failure, the database recovery process has no data on the contents of the row cache. Therefore, it does not repopulate the row caches in memory.

The CHECKPOINT clause of the CREATE CACHE, ADD CACHE, or ALTER CACHE clause overrides this database-level CHECKPOINT clause.

LOCATION IS directory-spec

This is a subclause of other clauses and has different effects, depending upon the clause in which it is used, as follows:

- In the row-cache-options clause

  Specifies the name of the default directory to which row cache backing file information is written. The database system generates a file name (row-cache-name.rdc) automatically for each row cache backing file it creates when the RCS process starts up. Specify a device name and directory name enclosed within single quotation marks('); do not include a file specification. The file name is the row-cache-name specified when creating the row cache. By default, the location is the directory of the database root file.

  The LOCATION clause of the CREATE CACHE, ADD CACHE, or ALTER CACHE clause overrides this location, which is the default for the database.

- In a CREATE CACHE, ADD CACHE, or ALTER CACHE clause (row-cache-params1 clause)

Specifies the name of the directory to which row cache backing file information is written. The database system generates a file name (row-cache-name.rdc) automatically for each row cache at checkpoint time. Specify a device name and directory name enclosed within single quotation marks ('); do not include a file specification. The file name is the row cache name specified when creating the row cache. By default, the location is the directory of the database root file. These .rdc files are permanent database files.

This LOCATION clause overrides a previously specified location at the database level (in a row-cache-options clause).

This clause is ignored if the row cache is defined to checkpoint to the database.

NO LOCATION

This is a subclause of other clauses and has different effects, depending upon the clause in which it is used, as follows:

- In the row-cache-options clause

  Removes the location previously specified in a LOCATION IS clause for the row cache. If you specify NO LOCATION, the row cache location becomes the directory of the database root file.

  The LOCATION clause of the CREATE CACHE, ADD CACHE, or ALTER CACHE clause overrides this location, which is the default for the database.

- In a CREATE CACHE, ADD CACHE, or ALTER CACHE clause (row-cache-params1 clause)

  Removes the location previously specified in a LOCATION IS clause for the row cache backing file. If you specify NO LOCATION, the row cache location becomes the directory of the database root file.

  This clause is ignored if the row cache is defined to checkpoint to the database.

RECOVERY JOURNAL (LOCATION IS directory-spec)

Specifies the location, including device and directory, in which the recovery-unit journal (.ruj) file is written. Do not include network node names, file names or process-concealed logical names.

If this clause is omitted, the default directory location is the current device the database root file uses and the special directory [RDM$RUJ]. You can use the RDMS$RUJ logical name to override this clause.

RECOVERY JOURNAL (NO LOCATION)

Removes a location previously defined by a RECOVERY JOURNAL LOCATION IS clause. This causes the recovery journal to revert to the default location.

SHARED MEMORY IS SYSTEM
SHARED MEMORY IS PROCESS

Determines whether cache global sections are created in system space or process space.

When you use cache global sections created in the process space, you and other users share physical memory, and the OpenVMS Alpha operating system maps a row cache area to a private address space for each user. As a result, all users are limited by the free virtual address range, and each uses a percentage of memory in overhead. If many users are accessing the database, the overhead can be high.

When many users are accessing the database, consider using SHARED MEMORY IS SYSTEM. This gives users more physical memory because they share the system space of memory and there is none of the overhead associated with the process space of memory.

The default is SHARED MEMORY IS PROCESS.

SHARED MEMORY IS PROCESS RESIDENT

The SHARED MEMORY clause was previously available with only the SYSTEM and PROCESS options. These options determine whether database root global sections (including global buffers when enabled) or whether the cache global sections are created in system space or process space. The RESIDENT option extends the PROCESS option by making the global section memory resident.

SECURITY CHECKING

Traditionally Oracle Rdb has performed security checking using the operating system security layer (for example, the UIC and rights identifiers of the OpenVMS operating system).

The access control list (ACL) information stored in the database contains a granted privilege mask and a set of users represented by a unique integer (for example, a UIC).

There are three modes of security checking:

- SECURITY CHECKING IS EXTERNAL

  This is the default. External security checking recognizes database users (created with the SQL CREATE USER statement) as operating system user identification codes (UICs) and roles as special rights identifiers or groups. PERSONA support is enabled or disabled as follows:

- SECURITY CHECKING IS EXTERNAL (PERSONA SUPPORT IS ENABLED)

  Enables the full impersonation of an OpenVMS user. This means the UIC and the granted right identifiers are used to check access control list permissions.

  Note: If PERSONA is enabled, you cannot connect to an Oracle Rdb database through Oracle SQL/Services. Impersonation is not supported in Oracle SQL/Services.

- SECURITY CHECKING IS EXTERNAL (PERSONA SUPPORT IS DISABLED)

  Disables the full impersonation of an OpenVMS user. Only the UIC is used to check access control list permissions. This is the default for a new database, or for a database converted from a prior version of Oracle Rdb.

- SECURITY CHECKING IS INTERNAL (ACCOUNT CHECK IS ENABLED)

  The ACCOUNT CHECK clause ensures that Oracle Rdb validates the current database user with the user name (such as defined with an SQL CREATE USER statement) stored in the database. This prevents different users with the same name from accessing the database. Therefore, this clause might prevent a breach in security.

  Note: The ACCOUNT CHECK IS ENABLED clause on OpenVMS does not check the SID, but does force the user session to have the same user name and UIC as recorded in the database.

  If you specify the ACCOUNT CHECK IS DISABLED clause, then a user with a matching UIC (also called a profile-id) is considered the same as the user even if his or her user name is different. This allows support for multiple OpenVMS users with the same UIC.

- SECURITY CHECKING IS INTERNAL (ACCOUNT CHECK IS DISABLED)

  This syntax means that the assigned SID is ignored during database attach.

SYNONYMS ARE ENABLED
Enables synonym creation. A synonym is a special object that provides an alternate name for an existing database object.

CACHE USING row-cache-name
Specifies that the named row cache is the default physical row cache for all storage areas in the database. All rows stored in each storage area are cached, regardless of whether they consist of table data, segemented string data, or are special rows (such as index nodes).

You must either add the specified cache before completing the ALTER DATABASE statement, or it must already exist.

Alter the database and storage area to assign a new physical area row cache that overrides the database default physical area row cache. Only one physical area row cache is allowed for each storage area.

You can have multiple row caches that contain rows for a single storage area by defining logical area row caches, where the row cache name matches the name of a table or index.

If you do not specify the CACHE USING clause or the NO ROW CACHE clause, then the NO ROW CACHE clause is the default.

NO ROW CACHE
Specifies that the database default is to not assign a row cache to all storage areas in the database. You cannot specify the NO ROW CACHE clause if you specify the CACHE USING clause.

Alter the storage area and name a row cache to override the database default. Only one row cache is allowed for each storage area.

If you do not specify the NO ROW CACHE clause or the CACHE USING clause, then the NO ROW CACHE clause is the default.

add-row-cache-clause
Creates a new row cache.

ALLOCATION IS n BLOCK
ALLOCATION IS n BLOCKS
Specifies the initial allocation of the row cache backing file (.rdc) to which cached rows are written during a checkpoint operation.

If the ALLOCATION clause is not specified, the default allocation in blocks is approximately 40 percent of the CACHE SIZE for this row cache.

This clause is ignored if the row cache is defined to checkpoint to the database.

EXTENT IS n BLOCK
EXTENT IS n BLOCKS
Specifies the file extent size for the row cache backing file (.rdc).

If the EXTENT clause is not specified, the default number of blocks is equal to the CACHE SIZE * 127 for the cache being added.

This clause is ignored if the row cache is defined to checkpoint to the database.

CACHE SIZE IS n ROW
CACHE SIZE IS n ROWS
Specifies the number of rows allocated to the row cache. As the row cache fills, the most recently referenced rows are retained in the row cache, while those not referenced recently are discarded. Adjusting the allocation of the row cache helps to retain important rows in memory. If the CACHE SIZE clause is not specified, the default is 1000 rows.

The product of the CACHE SIZE and the ROW LENGTH settings determines the amount of memory required for the row cache. (Some additional overhead and rounding up to page boundaries are performed by the database system.) The row cache is shared by all processes attached to the database.

LARGE MEMORY IS ENABLED
LARGE MEMORY IS DISABLED
Specifies whether or not large memory is used to manage the row cache. Very large memory (VLM) allows Oracle Rdb to use as much physical memory as is available. It provides access to a large amount of physical memory through small virtual address windows.

Use the LARGE MEMORY IS ENABLED clause only when both of the following are true:

- You have enabled row caching.
- You want to cache large amounts of data, but the cache does not fit in the virtual address space.

The default is the LARGE MEMORY IS DISABLED clause.

ROW REPLACEMENT IS ENABLED
ROW REPLACEMENT IS DISABLED
Specifies whether or not Oracle Rdb replaces rows in the cache. When the ROW REPLACEMENT IS ENABLED clause is used, rows are replaced when the row

cache becomes full. When the ROW REPLACEMENT IS DISABLED clause is used, rows are not replaced when the cache is full. The type of row replacement policy depends upon the application requirements for each cache.

The default is the ROW REPLACEMENT IS ENABLED clause.

NUMBER OF SWEEP ROWS IS n

Specifies the number of modified cache rows that will be written back to the database when a sweep operation occurs to make space available in the cache for subsequent transactions to insert rows into the cache. Oracle Corporation recommends that you initially specify the number of sweep rows to be between 10 and 30 percent of the total number of rows in the cache. Then monitor performance and adjust the number of sweep rows, if necessary. The default setting is 3000 rows.

NUMBER OF RESERVED ROWS IS n

Specifies the maximum number of cache rows that each user can reserve. The default is 20 rows.

This value is also used when searching for available slots in a row cache. The entire row cache is not searched on the initial pass. This value specifies the maximum number of rows that are searched for a free slot. If at least one free slot is found, the insert operation can proceed. If no free slots are found in this initial search, Oracle Rdb continues searching through the cache until it finds a free slot.

ROW LENGTH IS n BYTE
ROW LENGTH IS n BYTES

Specifies the size of each row allocated to the row cache. Rows are not cached if they are longer than a row cache row. The row length is an aligned longword rounded up to the next multiple of 4 bytes.

The default row length for a table is derived from the table row length and the default row length for a sorted index is derived from the index node size. For all other objects, the default row length is 256 bytes.

WINDOW COUNT IS n

Specifies the number of virtual address windows used by the LARGE MEMORY clause.

The window is a view into the physical memory used to create the very large memory (VLM) information. Because the VLM size may be larger than that which can be addressed by a 32-bit pointer, you need to view the VLM information through small virtual address windows.

You can specify a positive integer in the range from 10 through 65535.

The default is WINDOW COUNT IS 100.

ALTER CACHE row-cache-name
Alters an existing row cache.

DROP CACHE row-cache-name CASCADE
DROP CACHE row-cache-name RESTRICT
Deletes the specified row cache from the database. If the mode is RESTRICT, then an exception is raised if the row cache is assigned to a storage area. If the mode is CASCADE, then the row cache is removed from all referencing storage areas.

The default is RESTRICT if no mode is specified.

Usage Notes

- The syntax NUMBER OF CLUSTER NODES IS 1 (MULTIPLE INSTANCE) is contradictory and causes the CREATE DATABASE, ALTER DATABASE, or IMPORT statement to fail.
- When SINGLE INSTANCE is enabled, the PAGE TRANSFER VIA MEMORY attribute of the GLOBAL BUFFERS ARE ENABLED clause is permitted when the number of cluster nodes exceeds 1.
- The SINGLE INSTANCE and MULTIPLE INSTANCE options are not permitted for the deprecated NUMBER OF VAXCLUSTER NODES clause.
- The PRESTARTED TRANSACTION attribute in the database will be used unless overridden by the RDMS$PRESTART_TXN logical name, or the PRESTARTED TRANSACTION clause on an explicit ATTACH, CONNECT, or DECLARE ALIAS statement. However, the time value specified for the database is used if prestarted transactions are enabled.
- To enable or disable SHARED MEMORY IS PROCESS RESIDENT, the process executing the command must be granted the VMS$MEM_RESIDENT_USER rights identifier. When this feature is enabled, the process that opens the database must also be granted the VMS$MEM_RESIDENT_USER rights identifier. Oracle Corporation recommends using the RMU Open command when utilizing this feature.
- To enable or disable Galaxy support, the process executing the command must hold the SHMEM privilege. When Galaxy is enabled, the process that opens the database must have the SHMEM privilege enabled. Oracle Corporation recommends using the RMU Open command when utilizing this feature.
- The number of reserved slots for sequences cannot be decreased.

- If you do not specify the RESERVE n CACHE SLOTS clause, the default number of cache slots is 32.
- See Chapter 4 for information and guidance on using row caching.
- The following usage notes apply to the DROP STORAGE AREA CASCADE clause:
  - Beginning with this release, an index that is not partitioned and resides entirely in the storage area being dropped will be deleted using CASCADE semantics (and therefore will invalidate any query outlines that refer to that index). Prior to Oracle Rdb release 7.1, the statement would fail when such an index resided in the storage area being dropped.
  - The NOT NULL, PRIMARY KEY, and UNIQUE constraints for affected tables are ignored by the DROP STORAGE AREA CASCADE clause because validation of these constraints is not necessary.

    These types of constraints are not affected by removal of rows from the table. However, CHECK and FOREIGN constraints on the affected table and referencing tables are still evaluated.
  - When DROP STORAGE AREA CASCADE is executing, it logs debugging messages to the standard output device or the RDMS$DEBUG_FLAGS_OUTPUT log file, if defined.

    You can enable logging of the debug messages using the logical name RDMS$SET_FLAGS, which accepts the same input as the SQL SET FLAGS statement. For example:

    ```
    $DEFINE RDMS$SET_FLAGS 'STOMAP_STATS, INDEX_STATS, ITEM_LIST'
    ```

    The SET FLAGS options shown in the preceding example enables the following debug output:

    * STOMAP_STATS displays the processing of storage maps for any tables that refer to the dropped storage area. The output is prefixed with "~As". This has the same effect as setting the RDMS$DEBUG_FLAGS logical name to "As".
    * INDEX_STATS displays the processing of any indexes that reference the dropped storage area. The output is prefixed with "~Ai". This has the same effect as setting the RDMS$DEBUG_FLAGS logical name to "Ai".
    * ITEM_LIST displays the names of any constraints that require processing. This has the same effect as setting the RDMS$DEBUG_FLAGS logical name to "H".

The output includes the discovered tables and indexes, some decision-point information (does an index need to be deleted, does a partition need to be scanned, and so on), and I/O statistics for the storage map pruning operations.

Part of the DROP STORAGE AREA CASCADE operation may include deleting tables and indexes. These are processed internally as DROP TABLE CASCADE and DROP INDEX CASCADE operations. However, by the time these commands execute, all references to the dropped storage area will have been removed. Therefore, in many cases, the DROP TABLE or DROP INDEX statement only cleans up the metadata definition; there is no need to scan the storage area.

– The time required to delete a storage area file depends on the size of the directory file, the file allocation, and the number of extents made by the file system to grow the file. If the ERASE ON DELETE attribute is enabled on the disk, then this must also be factored into the time calculations (allow time for the file system to overwrite the file with an erase pattern).

– The read/write I/O statistics are output only if the database has statistics collection enabled. Statistics collection might be disabled if the logical name RDM$BIND_STATS_ENABLED has been set to 0 or if an ALTER DATABASE ... STATISTICS COLLECTION IS DISABLED statement has been issued.

Examples

Example 1 Establishing a Timeout Value for Prestarted Transactions

```
SQL> ALTER DATABASE
cont>   FILENAME SAMPLE
cont>   PRESTARTED TRANSACTIONS ARE ENABLED
cont>      (WAIT 90 SECONDS FOR TIMEOUT)
cont>   ;
```

Example 2 Altering a Database Specifying the SINGLE INSTANCE Option

```
SQL> ALTER DATABASE FILENAME PERSONNEL
cont> NUMBER OF CLUSTER NODES IS 4 (SINGLE INSTANCE);
```

Example 3 Altering a Database Specifying the PROCESS RESIDENT Option

```
SQL> ALTER DATABASE FILENAME PERSONNEL
```

```
cont> SHARED MEMORY IS PROCESS RESIDENT;
```

Example 4 Reserving Slots for Sequences

```
SQL> ALTER DATABASE FILENAME mf_personnel
cont> RESERVE 10 SEQUENCES;
SQL> EXIT
$ RMU/DUMP/HEADER=BRIEF mf_personnel.rdb
.
.
Client sequences:
    - 64 client sequences have been allocated.
.
.
.
SQL> ALTER DATABASE FILENAME mf_personnel
cont> RESERVE 33 SEQUENCES;
SQL> EXIT
$ RMU/DUMP/HEADER=BRIEF mf_personnel.rdb
.
.Client sequences:
    - 128 client sequences have been allocated.
```

This example demonstrates how to reserve slots for sequences. When a database is created, 32 sequences are reserved by default. This example shows that additional sequences are reserved in increments of 32. If you reserve less than a multiple of 32, 32 slots are reserved.

Example 5 Adding and Enabling a Row Cache on OpenVMS

```
SQL> ALTER DATABASE FILENAME MF_PERSONNEL
cont> NUMBER OF CLUSTER NODES IS 1
cont> JOURNAL ENABLED (FAST COMMIT ENABLED)
cont> RESERVE 20 CACHE SLOTS
cont> ROW CACHE IS ENABLED
cont> ADD CACHE EMPIDS_LOW_RCACHE
cont> SHARED MEMORY IS PROCESS
cont> ROW LENGTH IS 104 BYTES
cont> CACHE SIZE IS 204 ROWS
cont> CHECKPOINT ALL ROWS TO BACKING FILE
cont> NUMBER OF SWEEP ROWS IS 25
cont> LOCATION IS 'USER2:[RCS]'
cont> ALLOCATION IS 1 BLOCK;
%RDMS-W-DOFULLBCK, full database backup should be done to ensure future recovery
SQL>
```

ALTER DOMAIN Statement

Alters a domain definition. A **domain** is the set of values that a column in a table can have. A domain definition specifies the set of values by associating an SQL data type with a domain name. The CREATE and ALTER TABLE statements can use the domain in column definitions.

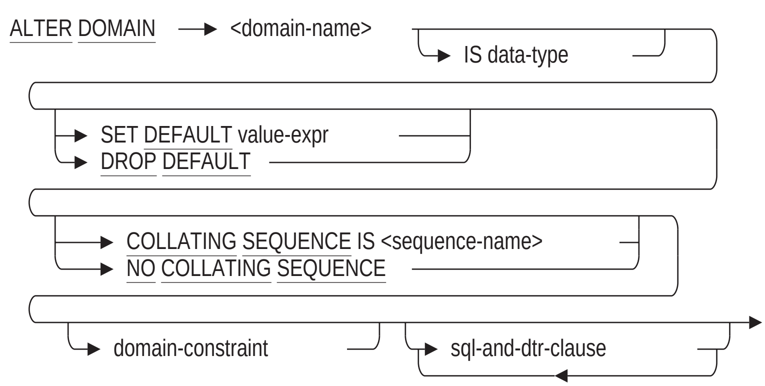
The ALTER DOMAIN statement lets you change the character set, data type, optional default value, optional collating sequence, or optional formatting and DATATRIEVE clauses associated with a domain name. Any table definitions that refer to that domain reflect the changes.

Environment

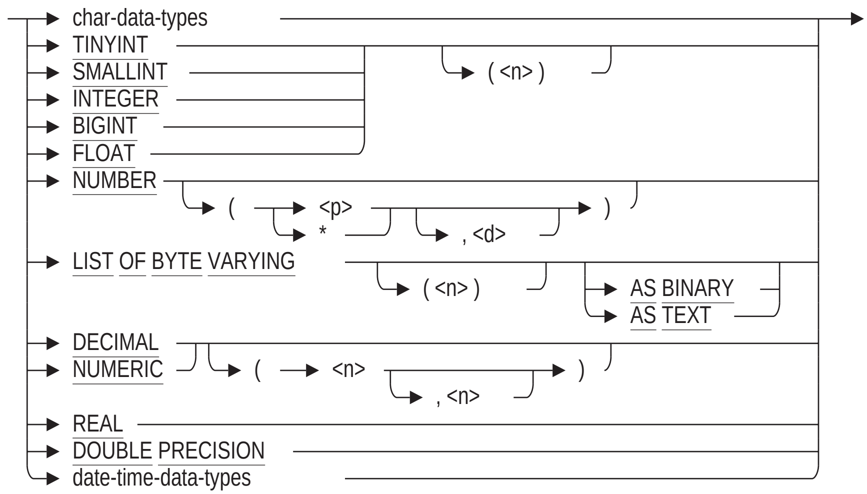
You can use the ALTER DOMAIN statement:

- In interactive SQL
- Embedded in host language programs to be precompiled
- As part of a procedure in an SQL module
- In dynamic SQL as a statement to be dynamically executed

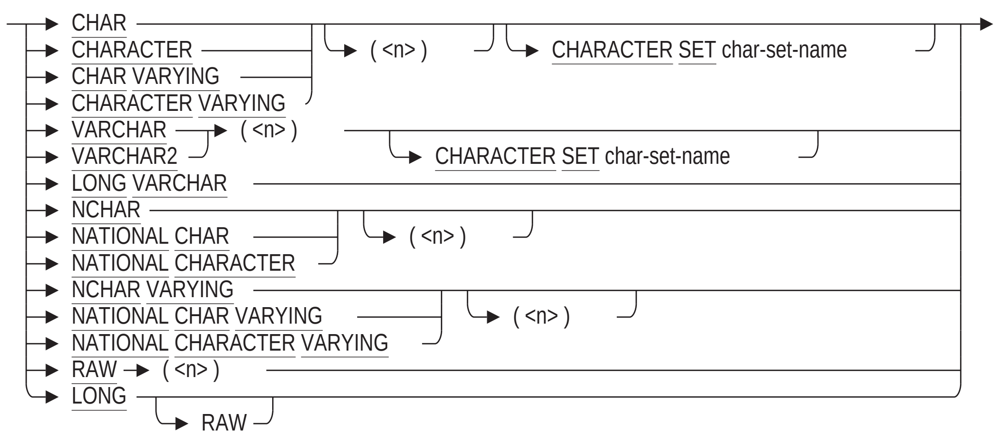
Format



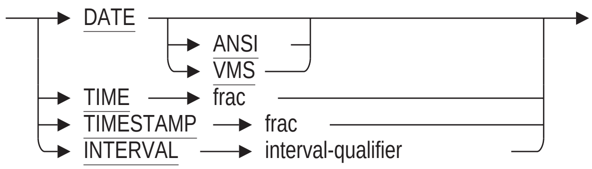
data-type =



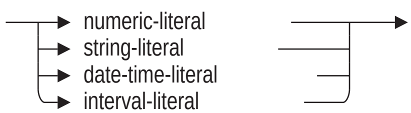
char-data-types =



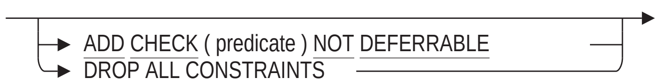
date-time-data-types =



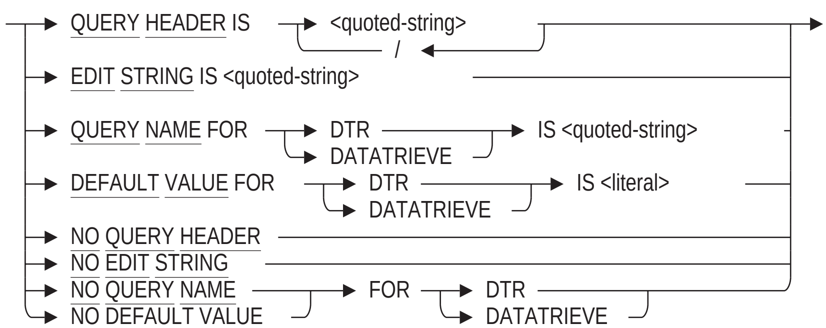
literal =



domain-constraint =



sql-and-dtr-clause =



Arguments

value-expr
Specifies the default value of a domain.

Usage Notes

No new usage notes.

Examples

Example 1 Altering a Domain to Provide a Default Value

```
SQL> -- Display the current domain definition.
SQL> SHOW DOMAIN DEPARTMENT_NAME
DEPARTMENT_NAME                  CHAR(30)
 Comment:        Department name
 Missing Value: None
SQL> -- Alter the domain to provide a default value
SQL> -- for DEPARTMENT_NAME.
SQL> ALTER DOMAIN DEPARTMENT_NAME
cont> SET DEFAULT 'Not Recorded';
SQL> -- Display the altered domain definition.
SQL> SHOW DOMAIN DEPARTMENT_NAME;
DEPARTMENT_NAME                  CHAR(30)
 Comment:        Department name
 Oracle Rdb default: Not Recorded
 Missing Value: None
SQL> -- Insert a record and omit the value for DEPARTMENT_NAME.
SQL> INSERT INTO DEPARTMENTS (DEPARTMENT_CODE)
cont> VALUES
cont> ('GOGO');
1 row inserted
SQL> COMMIT;
SQL> -- Select the newly inserted record to show that the
SQL> -- default for the DEPARTMENT_NAME domain was inserted.
SQL> SELECT * FROM DEPARTMENTS WHERE DEPARTMENT_CODE='GOGO';
 DEPARTMENT_CODE   DEPARTMENT_NAME                   MANAGER_ID
   BUDGET_PROJECTED   BUDGET_ACTUAL
 GOGO              Not Recorded                      NULL
              NULL            NULL
1 row selected
```

ALTER FUNCTION Statement

Allows a comment to be changed for a function that was created in one of the following ways:

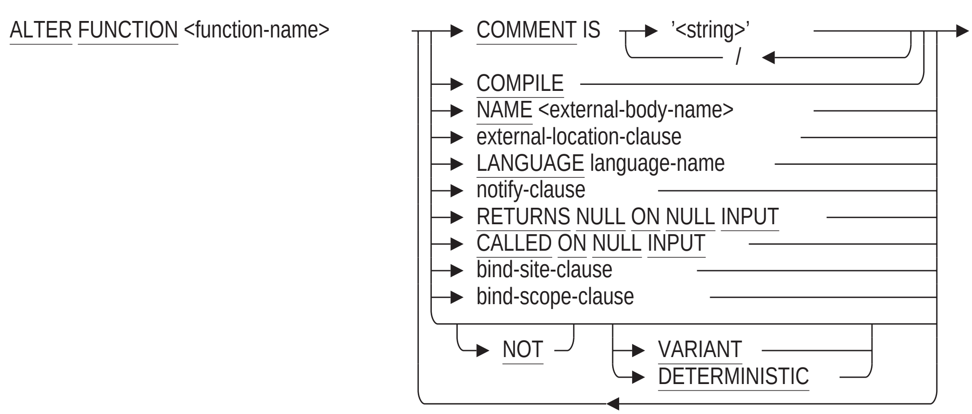
- Using the CREATE MODULE statement or the CREATE FUNCTION statement (COMMENT IS clause)
- Using the COMMENT ON FUNCTION statement
- Forcing a stored (SQL) function to be compiled (COMPILE option)
- Modifying attributes of external functions

Environment

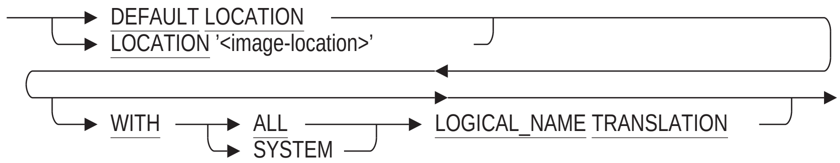
You can use the ALTER FUNCTION statement:

- In interactive SQL
- Embedded in host language programs
- As part of a procedure in an SQL module or other compound statement
- In dynamic SQL as a statement to be dynamically executed

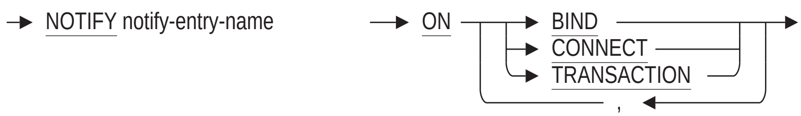
Format



external-location-clause =

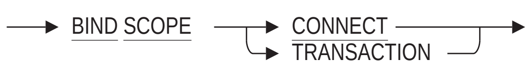


notify-clause =



bind-site-clause =

bind-scope-clause =



Arguments

COMMENT IS 'string'

Adds a comment about the function or procedure parameters. SQL displays the text of the comment when it executes a SHOW FUNCTIONS or SHOW PROCEDURES statement. Enclose the comment in single quotation marks (') and separate multiple lines in a comment with a slash mark (/).

RETURNS NULL ON NULL INPUT
CALLED ON NULL INPUT

These clauses control how an external function is invoked when one or more of the function arguments is NULL. The CALLED ON NULL INPUT clause specifies that the function should be executed normally. A normal execution when the PARAMETER STYLE GENERAL clause is specified means that SQL should return a run-time error when the NULL value is detected.

The RETURNS NULL ON NULL INPUT clause instructs Oracle Rdb to avoid the function call and just return a NULL result. This option is valuable for library functions such as SIN, COS, CHECKSUM, SOUNDEX, and so on, that usually return an UNKNOWN result if an argument is NULL.

The CALLED ON NULL INPUT clause is the default.

DETERMINISTIC
NOT DETERMINISTIC

These clauses are synonyms for the VARIANT and NOT VARIANT clauses for conformance to the SQL/PSM standard.

The DETERMINISTIC clause indicates that the same inputs to the function will generate the same output. It is the same as the NOT VARIANT clause.

The NOT DETERMINISTIC clause indicates that the output of the function does not depend on the inputs. It is the same as the VARIANT clause.

COMPILE

The COMPILE option forces the Oracle Rdb server to recompile the stored (SQL) routine. External routines are not affected.

Use COMPILE when a routine has been made invalid by the execution of a DROP ... CASCADE operation. This mechanism is preferred over the SET FLAGS 'VALIDATE_ROUTINE' method available in previous versions.

Usage Notes

- The ALTER FUNCTION statement causes the RDB$LAST_ALTERED column of the RDB$ROUTINES table for the named function to be updated with the transactions time stamp.
- All of the arguments for the ALTER FUNCTION statement, with the exception of the COMPILE, COMMENT, VARIANT, and DETERMINISTIC arguments, apply only to external functions. An error is returned if the referenced function is an SQL stored function.

ALTER INDEX Statement

Changes an index. The ALTER INDEX statement allows you to:

- Change the characteristics of index nodes (sorted indexes only)
- Change the names of the storage areas that contain the index
- Enable or disable logging to the .aij and .ruj files
- Add a partition
- Drop a partition
- Move a partition
- Change a partition name
- Change the description of a partition
- Specify whether or not the index is UNIQUE

You cannot change:

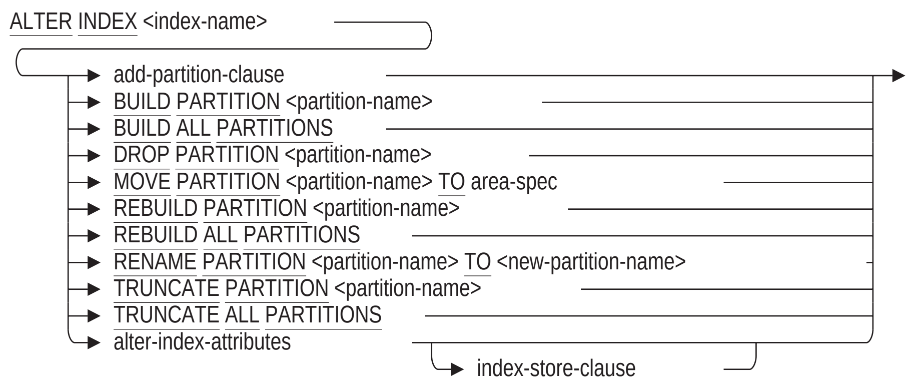
- The columns that comprise an index
- A hashed index to a sorted index
- A sorted index to a hashed index
- A sorted, nonranked index to a sorted, ranked index
- A sorted, ranked index to a sorted, nonranked index
- The duplicates compression of a sorted, ranked index

Environment

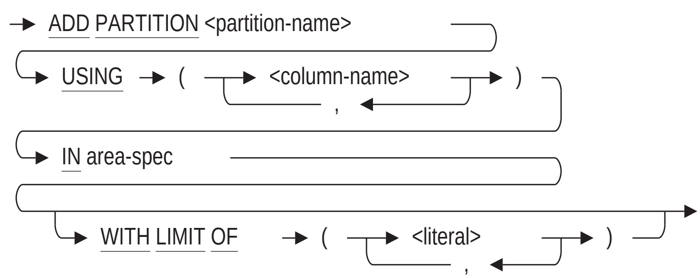
You can use the ALTER INDEX statement:

- In interactive SQL
- Embedded in host language programs to be precompiled
- As part of a procedure in an SQL module
- In dynamic SQL as a statement to be dynamically executed

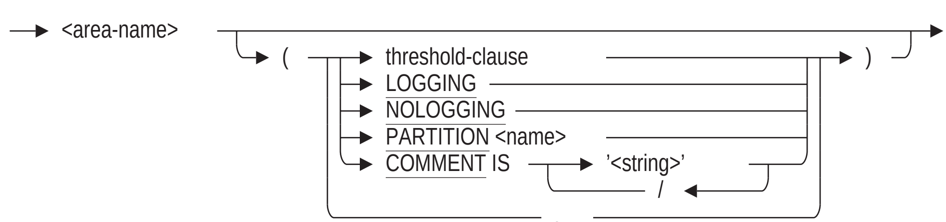
Format



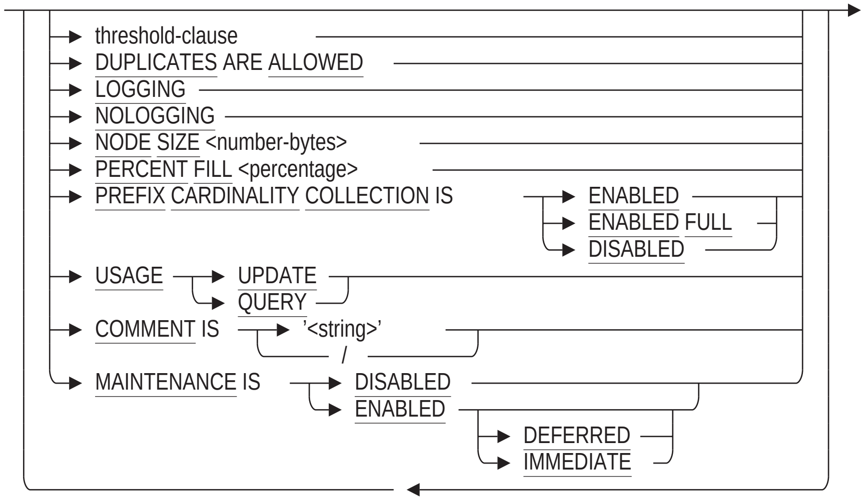
add-partition-clause =



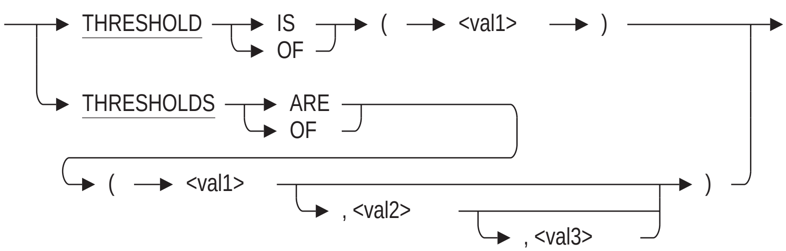
area-spec =



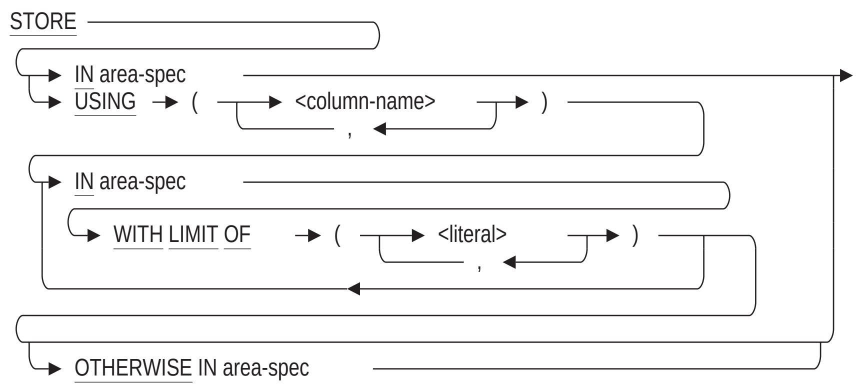
alter-index-attributes=



threshold-clause =



index-store-clause =



Arguments

ADD PARTITION partition-name

Adds the named partition to an existing hashed index. The partition name must be unique within the index being altered. See the Usage Notes for additional information about using the ADD PARTITION clause.

USING (column-name)

Specifies columns whose values are used as limits for partitioning the index across multiple storage areas. You cannot name columns not specified as index key segments.

If the index key is multisegmented, you can include some or all of the columns that are joined to form the index key. You must specify the columns in the order in which they were specified when the index key was defined. If you include only a subset of the columns from the multisegmented index, you must include the leading columns of the index key.

IN area-spec
To area-spec

When specified as part of an ADD PARTITION clause, the IN area-spec creates a new storage area for the partition. If you do not specify a WITH LIMIT OF clause or OTHERWISE clause, the IN area-spec clause creates a new final partition.

When specified as part of an index-store-clause, the IN area-spec clause associates the index directly with a single storage area, and all entries in the index are stored in the area you specify.

WITH LIMIT OF (literal)

Specifies the highest value for the index key that resides in a particular storage area if ASCENDING is defined. If DESCENDING is defined, the lowest value is specified for the index key that resides in a particular storage area. For multicolumn index keys, specify a literal value for each column listed in the USING clause.

The WITH LIMIT OF clause must specify a new unique set of values for the partition. The number of literals in the list must be the same as the number of columns in the USING clause. The data type of the literals must agree with the data type of the column. For character columns, enclose the literals in single quotation marks.

BUILD PARTITION <partition-name>

This clause operates on an index in build-pending state (created using the MAINTENANCE IS ENABLED DEFERRED clause) and builds the named partition. If the index is not in build-pending state, then the statement completes successfully with a warning.

No other clauses may appear in the same ALTER INDEX statement.

BUILD ALL PARTITIONS

This clause operates on an index in build-pending state (created using the MAINTENANCE IS ENABLED DEFERRED clause) and builds all incomplete partitions. If the index is not in build-pending state, then the statement completes successfully with a warning.

No other clauses may appear in the same ALTER INDEX statement.

DROP PARTITION partition-name

Specifies that the data in the named partition be migrated to the next partition in the map and the named partition be dropped. The last partition in the index cannot be dropped.

PARTITION name

Names the partition. The name can be a delimited identifier if the dialect is set to SQL99, SQL92, or ORACLE LEVEL1 or quoting rules are set to SQL92 or SQL99. Partition names must be unique within the index. If you do not specify this clause, Oracle Rdb generates a default name for the partition. The partition name is stored in the database and validated (prior to Oracle Rdb release 7.1, this was not done).

MOVE PARTITION partition-name TO area-spec
Specifies that the data in the named partition be moved to the partition identified in the area-spec clause and that the current partition name of the data be dropped after the data is migrated. For example, this clause allows a single hashed index partition to be moved to a larger storage area when too many mixed area extends are observed.

threshold-clause
See the CREATE INDEX Statement for a description of the threshold-clause.

COMMENT IS 'string'
Adds a comment about the storage map definition for the index. SQL displays the text of the comment when it executes a SHOW INDEX statement. Enclose the comment in single quotation marks (') and separate multiple lines in a comment with a slash mark (/).

LOGGING
NOLOGGING
The LOGGING clause specifies that updates to new index partitions should be logged in the recovery-unit journal file (.ruj) and after-image journal file (.aij).

The NOLOGGING clause specifies that updates to new index partitions should not be logged in the recovery-unit journal file (.ruj) and after-image journal file (.aij).

If no STORE clause is used, then these attributes provide the setting for the ALTER INDEX statement.

The LOGGING and NOLOGGING clauses are mutually exclusive; specify only one. The LOGGING clause is the default.

See Section 2.2.8 for information on the advantages and disadvantages of specifying these clauses and the implications that using the NOLOGGING keyword has on database recovery.

REBUILD PARTITION <partition-name>
This clause combines the TRUNCATE and BUILD actions into a single function for the named partition. No other clauses may appear in the same ALTER INDEX statement.

REBUILD ALL PARTITIONS
This clause combines the TRUNCATE and BUILD actions into a single function. No other clauses may appear in the same ALTER INDEX statement.

RENAME PARTITION partition-name TO new-partition-name
Changes the name of a partition. This clause can be applied to all types of indexes. It is particularly useful for specifying a more meaningful name for the default partition. Use the RMU Extract command to display the name of the default partition. See Example 3 in the Examples section.

TRUNCATE PARTITION <partition-name>
This clause operates on just the named index partition. The index is automatically set to MAINTENANCE IS ENABLED DEFERRED (that is, a build-pending state) if it was currently ENABLED IMMEDIATE. Otherwise, it stays in a disabled state.

No other clauses may appear in the same ALTER INDEX statement.

TRUNCATE ALL PARTITIONS
This clause operates in a similar way to TRUNCATE TABLE, but just on one index. The index is automatically set to MAINTENANCE IS ENABLED DEFERRED (that is, a build-pending state) if it was currently ENABLED IMMEDIATE. Otherwise, it stays in a disabled state.

No other clauses may appear in the same ALTER INDEX statement.

DUPLICATES ARE ALLOWED
Converts a UNIQUE index to a non-unique index. An index altered in this manner allows duplicate key values into the index. The only way to reverse this change once you commit the ALTER INDEX statement is to drop and redefine the index.

Usage Notes

- Note the difference between a partition and a storage area. A **partition** is a logical entity and a **storage area** is a physical entity. A table or index partition resides in a single storage area, but many such partitions from other tables and indexes can coexist in the same storage area. When a partition is said to be deleted, it is the logical entity within the storage area that is deleted. You move the data explicitly using the MOVE PARTITION clause or implicitly using the DROP PARTITION clause and then the ADD PARTITION clause.

  To drop the physical storage area file, use the ALTER DATABASE statement with the DROP STORAGE AREA clause.

- When the ALTER INDEX statement is used to change the partitioning of an index, it scans each referenced storage area to initialize and create the index partitions.

  The amount of I/O has been reduced (especially for mixed-format areas) by avoiding one of the area scans. Prior to this change, Oracle Rdb would scan each of the areas twice to remove the old index. Beginning with Oracle Rdb release 7.1, Oracle Rdb performs only a single scan of each area to remove the old hash index.

- The partition-name that you specify must be unique within the index being altered. The name is stored in the system table RDB$STORAGE_MAP_AREAS in the column RDB$PARTITION_NAME so that it can be used with other partition-related statements.
- Use the COMMENT ON statement to add comments to existing partitions. (See the COMMENT ON Statement for syntax.) This statement can be applied to all types of indexes.

  The partition-name must exist in the index being referenced. The name is checked against the system table RDB$STORAGE_MAP_AREAS and the column RDB$PARTITION_NAME. The old comment (if one exists), is deleted and replaced with the new text you specify and becomes permanent when a COMMIT statement is executed.
- If the INDEX_STATS flag is enabled, then the ALTER INDEX command logs messages to the file specified by the RDMS$DEBUG_FLAGS_OUTPUT logical name (or the standard output device, if the flag if not defined) to report the progress of an ADD, DROP, or MOVE PARTITION clause. The INDEX_STATS flag is enabled by doing one of the following:
  - Using the SET FLAGS 'INDEX_STATS' command.
  - Defining the RDMS$DEBUG_FLAGS logical to "Ai".
  - Defining the RDMS$SET_FLAGS logical name to 'INDEX_STATS'. See Example 2 in the Examples section.

  Note: The read / write I / O statistics shown in Example 2 in the Examples section are not displayed if statistics collection is disabled on the database or if the logical name RDM$BIND_STATS_ENABLED is defined as 0.

- The following notes apply to the ADD, DROP, and MOVE PARTITION clauses:
  - Currently, these clauses are supported for hashed indexes only. Support for sorted indexes will be provided in a future release.
  - When you add, drop, or move a partition, there must be no active queries compiled against the table. This includes declared cursors in the current

session and other applications that have referenced the table. As with other ALTER INDEX clauses, exclusive access to the table is required during the current transaction.

- The SHOW INDEX or SHOW TABLE (INDEX) command displays the original source of the index definition with the ADD, DROP, or MOVE PARTITION source appended. See Example 2 in the Examples section.

  Use the RMU Extract command with the Item=Index qualifier to see the current index definition with any additional partitions merged into the SQL CREATE INDEX syntax, dropped partitions omitted from the SQL CREATE INDEX syntax, or updated partitions (for moved partitions) in the SQL CREATE INDEX syntax.

- The following notes apply to the ADD PARTITION clause only:
  - The area-spec clause allows a PARTITION clause to name the partition. If you use the area-spec clause, the partition name it specifies must be the same as the partition name specified in the ADD PARTITION clause.
  - Oracle Rdb stores the partition name in the system table RDB$STORAGE_MAP_AREAS so that it can be used with other partition-related statements. The name is validated and must be unique per index.
  - To add partitions to an index, the index must have been created with a STORE clause so that additional partitions can be added.
  - The USING clause must list the same column names in the same order as in the original index definition.
  - If no WITH LIMIT OF clause is specified, then the partition is added at the end of the index as an otherwise partition. If an otherwise partition already exists for this index, then an error is reported.
  - The WITH LIMIT OF clause must specify a new unique set of values for the partition. A literal value must exist for each column listed in the USING clause.
  - When a new final partition or an otherwise partition is successfully added, no I/O to the index is required. That is, no data in the index needs to be relocated.

    The ADD PARTITION clause reads the RDB$SYSTEM_RECORD rows that are stored on each page of a mixed area and locates the hash buckets for the current index. Any hash keys that fall into the new partition are moved (with any associated duplicates) to the new partition. Any hash keys that do not belong in the newly added area are not moved.

Note: If this hashed index is used in a PLACEMENT VIA INDEX clause of a storage map, then those placed table rows are not moved by the ADD PARTITION clause. However, the new hashed index partition will correctly reference those rows even though they will no longer be stored adjacent to the hash bucket.

- If you attach to the database using the RESTRICTED ACCESS clause, then all partitions (and system record areas) are reserved for EXCLUSIVE access.

  These areas are also reserved for EXCLUSIVE access if the table appears in the RESERVING clause of the current transaction (either a DECLARE TRANSACTION or SET TRANSACTION statement) with an EXCLUSIVE mode.

  Otherwise, the default action is to reserve the new and subsequent partition of the index for PROTECTED WRITE. The RDB$SYSTEM_RECORD of the new partition is reserved for SHARED WRITE and the RDB$SYSTEM_RECORD of the existing partition is reserved for SHARED READ mode.

  Using EXCLUSIVE access to the partitions limits concurrent access to those storage areas by other users of the RDB$SYSTEM_RECORD, for instance, if other indexes are stored in that storage area. However, EXCLUSIVE access has the benefit of eliminating I/O to the associated snapshot files and reducing the virtual memory requirements of this operation. Therefore, Oracle Corporation recommends using EXCLUSIVE mode when possible to reduce the elapsed time of the ALTER INDEX operation. A COMMIT operation should be performed as soon as possible upon completion of the operation so that locks on the table are released.

  If the NOLOGGING clause is used (or the RDMS$CREATE_LAREA_NOLOGGING logical name is defined), then the hash buckets and duplicate nodes written to the new partition are not journaled. However, the updates to the existing RDB$SYSTEM_RECORD in that partition, and the deletions performed on the following partition, are journaled.

- The following notes apply to the DROP PARTITION clause only:
  - The partition-name must exist in the index being altered. The name is checked against the system table RDB$STORAGE_MAP_AREAS and the column RDB$PARTITION_NAME.
  - The index must have been created with a STORE clause, so that partitions can be referenced.

- There must be no active queries compiled against this table. This includes declared cursors in the current session, or other applications that have referenced the table. As with other ALTER INDEX statements, exclusive access to the table is required during the current transaction.
- The DROP PARTITION clause reads the RDB$SYSTEM_RECORD rows that are stored on each page of a mixed area and locates the hash buckets for the current index. All hash keys are moved (with any associated duplicates) to the next partition.

  Note: If this hashed index is used in a PLACEMENT VIA INDEX clause of a storage map, then those placed table rows are not moved by the DROP PARTITION clause. However, the hashed index will still correctly reference those rows even though they will no longer be stored adjacent to the hash bucket.

- If you attach to the database using the RESTRICTED ACCESS clause, then all partitions (and system record areas) will be reserved for EXCLUSIVE access.

  These areas will also be reserved for EXCLUSIVE access if the table appears in the RESERVING clause of the current transaction (either a DECLARE TRANSACTION or SET TRANSACTION statement) with an EXCLUSIVE mode.

  Otherwise, the default action is to reserve the old partition and its RDB$SYSTEM_RECORD for EXCLUSIVE access. The following partition is reserved for PROTECTED WRITE, and its RDB$SYSTEM_RECORD is reserved for SHARED WRITE.

  Using EXCLUSIVE access to the partitions will limit concurrent access to those storage areas by other users of the RDB$SYSTEM_RECORD, for instance, if other indexes are stored in that storage area. However, EXCLUSIVE access eliminates I/O to the associated snapshot files and reduces the virtual memory requirements of this operation. Therefore, Oracle Corporation recommends using EXCLUSIVE mode when possible to reduce the elapsed time of the ALTER INDEX operation. A COMMIT operation should be performed as soon as possible upon completion of the operation so that locks on the table are released.

■ The following notes apply to the MOVE PARTITION clause only:

– The partition-name must exist in the index being altered. The name is checked against the system table RDB$STORAGE_MAP_AREAS and the column RDB$PARTITION_NAME.

– The MOVE PARTITION clause can also rename the partition by including a PARTITION clause in the area-spec clause. The partition-name specified by this clause must have the same name as specified for the MOVE PARTITION or specify a unique name not already used by this index.

– The index must have been created with a STORE clause, so that partitions can be referenced.

– The MOVE PARTITION clause reads the RDB$SYSTEM_RECORD rows that are stored on each page of a mixed area and locates the hash buckets for the current index. All hash keys will be moved (with any associated duplicates) to the new storage area associated with this index.

Note: If this hashed index is used in a PLACEMENT VIA INDEX clause of a storage map, then those placed table rows are not moved by the MOVE PARTITION clause. However, the hashed index will still correctly reference those rows even though they will no longer be stored adjacent to the hash bucket.

– If you attach to the database using the RESTRICTED ACCESS clause, then all partitions (and system record areas) will be reserved for EXCLUSIVE access.

These areas will also be reserved for EXCLUSIVE access if the table appears in the RESERVING clause of the current transaction (either a DECLARE TRANSACTION or SET TRANSACTION statement) with an EXCLUSIVE mode.

Otherwise, the default action is to reserve the old partition and its RDB$SYSTEM_RECORD for EXCLUSIVE access. The following partition is reserved for PROTECTED WRITE, and its RDB$SYSTEM_RECORD is reserved for SHARED WRITE.

Using EXCLUSIVE access to the partitions will limit concurrent access to those storage areas by other users of the RDB$SYSTEM_RECORD, for instance, if other indexes are stored in that storage area. However, EXCLUSIVE access eliminates I/O to the associated snapshot files and reduces the virtual memory requirements of this operation. Therefore, Oracle Corporation recommends using EXCLUSIVE mode when possible to reduce the elapsed time of the ALTER INDEX operation. A COMMIT operation should be performed as soon as possible upon completion of the operation so that locks on the table are released.

– If the NOLOGGING clause is used (or if the RDMS$CREATE_LAREA_NOLOGGING logical name is defined) then the hash buckets and duplicate nodes written to the new partition are not journaled. However, the updates to the existing RDB$SYSTEM_RECORD in that partition, and the deletions performed on the following partition, are journaled.

- The following usage notes apply to the RENAME PARTITION clause only:
  – The partition-name must exist in the index being altered. The name is checked against the system table RDB$STORAGE_MAP_AREAS and the column RDB$PARTITION_NAME.
  – The new name replaces the current name and becomes permanent when a COMMIT statement is executed.
  – This clause can be applied to both sorted and hashed indexes.
- ALTER INDEX ... TRUNCATE PARTITION <partition-name> is ideal for large indexes that need to be deleted. It allows parts of the index to be deleted a little at a time. When DROP INDEX is finally used, the truncated partitions will not be reprocessed.
- The TRUNCATE TABLE statement will also truncate partitions of any build-pending indexes and change the state to enabled (because the empty index is complete for an empty table). Any index that has maintenance disabled will not be processed by the TRUNCATE TABLE statement.
- Oracle Corporation recommends using SET FLAGS with the INDEX_STATS option when using any of the following ALTER INDEX clauses:
  – BUILD PARTITION and BUILD ALL PARTITIONS
  – REBUILD PARTITION and REBUILD ALL PARTITIONS
  – TRUNCATE PARTITION and TRUNCATE ALL PARTITIONS

  The traced output describes the action of these clauses.

Examples

Example 1 Changing a Unique Index to Non-Unique

```
SQL> SHOW TABLE (INDEX) DEPARTMENTS;
Information for table DEPARTMENTS
```

```
Indexes on table DEPARTMENTS:
DEPARTMENTS_INDEX               with column DEPARTMENT_CODE
  No Duplicates allowed
  Type is Sorted
  Compression is DISABLED
Store clause:    WITHIN DEPARTMENTS
 Comment:        sorted index for departments

SQL> INSERT INTO DEPARTMENTS (DEPARTMENT_CODE) VALUES ('SUSO');
%RDB-E-NO_DUP, index field value already exists; duplicates not allowed for
 DEPARTMENTS_INDEX

SQL> ALTER INDEX DEPARTMENTS_INDEX DUPLICATES ARE ALLOWED;
This index was previously specified with a STORE clause. Continue? [N]Y
SQL> INSERT INTO DEPARTMENTS (DEPARTMENT_CODE) VALUES ('SUSO');
1 row inserted
SQL> SHOW TABLE (INDEX) DEPARTMENTS
Information for table DEPARTMENTS
Indexes on table DEPARTMENTS:
DEPARTMENTS_INDEX               with column DEPARTMENT_CODE
  Duplicates are allowed
  Type is Sorted
  Compression is DISABLED
Store clause:    WITHIN DEPARTMENTS
 Comment:        sorted index for departments
```

Example 2 Adding an Index Partition Before and After the Final Partition

```
SQL> CREATE UNIQUE INDEX EMPLOYEES_INDEX
cont>     ON EMPLOYEES (EMPLOYEE_ID)
cont>     TYPE IS HASHED
cont>     STORE USING (EMPLOYEE_ID)
cont>         IN JOBS WITH LIMIT OF ('00999');
SQL> COMMIT;
SQL> -- To add a partition before the final partition requires
SQL> -- that the final partition (which now follows the new partition)
SQL> -- be scanned and matching keys moved to the new partition.
SQL> SET TRANSACTION READ WRITE
cont>   RESERVING EMPLOYEES for EXCLUSIVE WRITE;
SQL> SET FLAGS 'INDEX_STATS';
SQL> ALTER INDEX EMPLOYEES_INDEX
cont>     ADD PARTITION NEW_EMPS_200
cont>       USING (EMPLOYEE_ID)
cont>       IN EMP_INFO WITH LIMIT OF ('00200');
```

```
~Ai alter index "EMPLOYEES_INDEX" (hashed=1, ordered=0)
~Ai add partition "NEW_EMPS_200" : area "EMP_INFO"
~Ai storage area "EMP_INFO" larea=85
~Ai splitting partition #1
~Ai split complete: total 100 keys, moved 37 (dups 0)
~Ai reads: async 136 synch 30, writes: async 57 synch 0
SQL> COMMIT;
SQL> -- Now add a partition after the final partition of
SQL> -- the index. This requires no I/O to the partition because
SQL> -- there is no following partition and therefore no keys
SQL> -- to be moved.
SQL> SET TRANSACTION READ WRITE
cont>   RESERVING EMPLOYEES FOR EXCLUSIVE WRITE;
SQL> ALTER INDEX EMPLOYEES_INDEX
cont>     ADD PARTITION NEW_EMPS_1400
cont>       USING (EMPLOYEE_ID)
cont>       IN EMPIDS_OVER WITH LIMIT OF ('01400');
~Ai alter index "EMPLOYEES_INDEX" (hashed=1, ordered=0)
~Ai add partition "NEW_EMPS_1400" : area "EMPIDS_OVER"
~Ai storage area "EMPIDS_OVER" larea=122
~Ai adding new final partition 3
SQL> COMMIT;
SQL> -- Show the index.  It shows the ADD PARTITION syntax appended
SQL> -- to the original source of the index.
SQL> SHOW INDEX EMPLOYEES_INDEX
Indexes on table EMPLOYEES:
EMPLOYEES_INDEX                  with column EMPLOYEE_ID
  No Duplicates allowed
  Type is Hashed Scattered
  Compression is DISABLED
Store clause:           STORE using (EMPLOYEE_ID)
                             in JOBS with limit of ('00999')
                        Add Partition partition NEW_EMPS_200
                        using (EMPLOYEE_ID)
                        in EMP_INFO with limit of ('00200')
                        Add Partition partition NEW_EMPS_1400
                        using (EMPLOYEE_ID)
                        in EMPIDS_OVER with limit of ('01400')
```

Example 3 Renaming a Partition

```
$ RMU/EXTRACT/ITEM=INDEX mf_personnel.rdb
.
.
```

```
.
create unique index EMPLOYEES_HASH
    on EMPLOYEES (
    EMPLOYEE_ID)
    type is HASHED
    store
        using (EMPLOYEE_ID)
            in EMPIDS_LOW(
                partition "SYS_P00076"
                )
                with limit of ('00200')
            in EMPIDS_MID(
                partition "SYS_P00077"
                )
                with limit of ('00400')
            otherwise in EMPIDS_OVER(
                partition "SYS_P00078"
                );
commit work;
C:\ORANT\mf_personnel>SQL
SQL> ATTACH 'FILENAME MF_PERSONNEL.RDB';
SQL> ALTER INDEX EMPLOYEES_HASH
cont> RENAME PARTITION SYS_P00076 TO IDS_LOW;
SQL> ALTER INDEX EMPLOYEES_HASH
cont> RENAME PARTITION SYS_P00077 TO IDS_MID;
SQL> ALTER INDEX EMPLOYEES_HASH
cont> RENAME PARTITION SYS_P00078 TO IDS_HIGH;
SQL> COMMIT;
SQL> SHOW INDEX EMPLOYEES_HASH;
Indexes on table EMPLOYEES:
EMPLOYEES_HASH                  with column EMPLOYEE_ID
  No Duplicates allowed
  Type is Hashed Scattered
  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
                        Rename PARTITION SYS_P00076 TO IDS_LOW
                        Rename PARTITION SYS_P00077 TO IDS_MID
                        Rename PARTITION SYS_P00078 TO IDS_HIGH
```

Example 4 Creating a Large Index Partitioned Across Twenty Storage Areas

First, create the database definition:

```
SQL> CREATE INDEX ... MAINTENANCE IS ENABLED DEFERRED ...;
```

Next submit twenty batch jobs to build each partition in parallel. For example, each batch job would execute a script similar to the following:

```
ATTACH 'filename testdatabase';
SET FLAGS 'index_stats';
ALTER INDEX TRANSACTIONS_INDEX BUILD PARTITION PART_1;
COMMIT;
```

Finally, after the batch jobs have completed, the database administrator must make the index active for query usage by changing the maintenance mode to ENABLED IMMEDIATE. A BUILD ALL PARTITIONS clause could be added in case any step failed (possibly due to resource limitations or a failed node).

```
SQL> SET FLAGS 'index_stats';
SQL> ALTER INDEX ... BUILD ALL PARTITIONS;
SQL> ALTER INDEX ... MAINTENANCE IS ENABLED IMMEDIATE;
SQL> COMMIT;
```

This scheme has several advantages over issuing a CREATE INDEX statement directly:

- The build actions can be run in parallel, which allows better resource usage (read and sort fewer rows), and reduced execution time for the index creation.
- The partitions being processed are relatively small when compared to the full index and, therefore, smaller quantities of data will be processed. This will result in smaller .ruj files and less AIJ file space for the transaction.
- Each build partition runs in a separate transaction, can easily be repeated if a step fails, and does not require repeating the entire CREATE INDEX statement.
- If any steps have failed, they will also be repeated by the BUILD ALL PARTITIONS clause included in the script.

Example 5 Deleting a Large Index Partitioned Across Twenty Storage Areas

First, disable the index:

```
SQL> ALTER INDEX TRANSACTIONS_INDEX MAINTENANCE IS DISABLED;
```

Next, submit twenty batch jobs to truncate the partitions in parallel:

```
SQL> ALTER INDEX TRANSACTIONS_INDEX TRUNCATE PARTITION PART_1;
```

Finally, after the batch jobs are complete, remove the metadata:

```
SQL> DROP INDEX TRANSACTIONS_INDEX;
```

This scheme has several advantages over a issuing a DROP INDEX statement directly:

- The truncate actions can be run in parallel, which allows better resource usage and reduced execution time for the index deletion.
- The partitions being processed are relatively small when compared to the full index and, therefore, smaller quantities of data will be processed. This will result in smaller .ruj files and less AIJ file space for the transaction.
- Each truncate partition runs in a separate transaction, can easily be repeated if a step fails, and does not require repeating the entire action.
- If any steps have failed, they will also be repeated by a DROP INDEX statement.

ALTER PROCEDURE Statement

Allows a comment to be changed for a procedure created in one of the following ways:

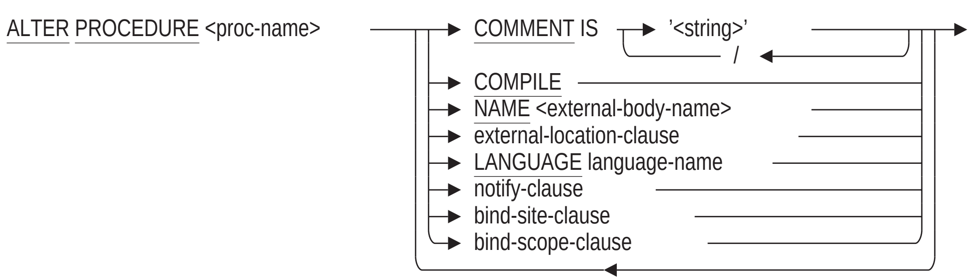
- Using the CREATE MODULE statement or CREATE PROCEDURE statement (COMMENT IS clause)
- Using the COMMENT ON PROCEDURE statement
- Forcing a stored (SQL) procedure to be compiled (COMPILE option)
- Modifying attributes of external procedures

Environment

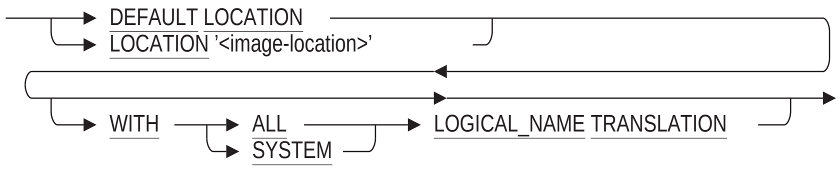
You can use the ALTER PROCEDURE statement:

- In interactive SQL
- Embedded in host language programs
- As part of a procedure in an SQL module or other compound statement
- In dynamic SQL as a statement to be dynamically executed

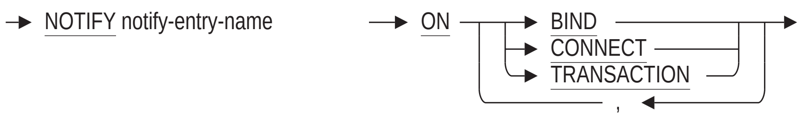
Format



external-location-clause =

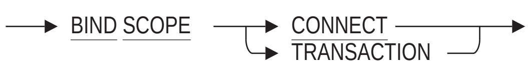


notify-clause =



bind-site-clause =

bind-scope-clause =



Arguments

COMMENT IS 'string'

Adds a comment about the function or procedure parameters. SQL displays the text of the comment when it executes a SHOW FUNCTIONS or SHOW PROCEDURES statement. Enclose the comment in single quotation marks (') and separate multiple lines in a comment with a slash mark (/).

COMPILE

The COMPILE option forces the Oracle Rdb server to recompile the stored (SQL) procedure. External procedures are not affected.

Use COMPILE when a routine has been made invalid by the execution of a DROP ... CASCADE operation. This mechanism is preferred over the SET FLAGS 'VALIDATE_ROUTINE' method available in previous versions.

Usage Notes

- The ALTER PROCEDURE statement causes the RDB$LAST_ALTERED column of the RDB$ROUTINES table for the named procedure to be updated with the transactions time stamp.
- All of the arguments for the ALTER PROCEDURE statement, with the exception of the COMPILE and COMMENT arguments, apply only to external procedures. An error is returned if the referenced routine is an SQL stored procedure.

ALTER PROFILE Statement

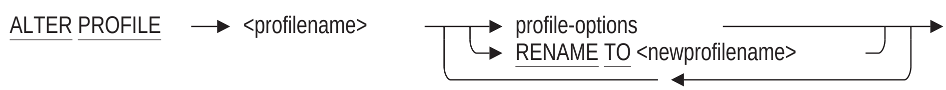
Alters a profile definition.

Environment

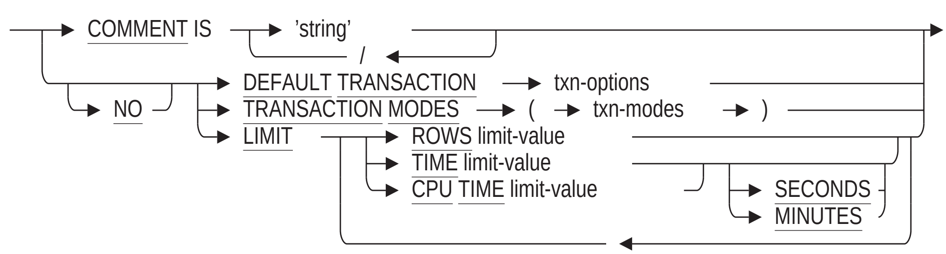
You can use the ALTER PROFILE statement:

- In interactive SQL
- Embedded in host language programs
- As part of a procedure in an SQL module
- In dynamic SQL as a statement to be dynamically executed

Format



profile-options =



limit-value =

Arguments

RENAME TO
This clause allows the database administrator to change the name of an existing profile.

See the description in CREATE PROFILE Statement for all other attributes supported by ALTER PROFILE.

NO DEFAULT TRANSACTION
NO TRANSACTION MODES
NO LIMIT CPU TIME
NO LIMIT ROWS
NO LIMIT TIME
These options explicitly record the negated attribute setting. These clauses will remove the current setting of any clause being negated.

Examples

The following example adds or changes a default transaction for an existing profile.

```
SQL> ALTER PROFILE DECISION_SUPPORT
cont>   DEFAULT TRANSACTION READ ONLY;
```

ALTER ROLE Statement

Allows you to change the role name or add a comment to a role.

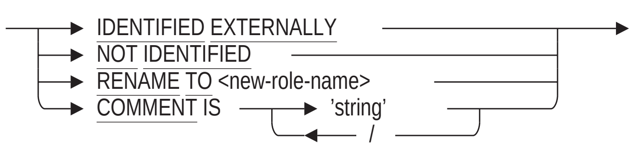
Environment

You can use the ALTER ROLE statement:

- In interactive SQL
- Embedded in host language programs to be precompiled
- As part of a procedure in an SQL module
- In dynamic SQL as a statement to be dynamically executed

Format

alter-role-opts =



Arguments

role-name
The name of an existing role (such as one created with the CREATE ROLE statement)

IDENTIFIED EXTERNALLY
NOT IDENTIFIED
Specifies whether SQL should inherit roles from the operating system. If you specify one of these clauses, you must specify the same clause as was specified

when the role was created. You cannot use the ALTER ROLE statement to change roles from IDENTIFIED EXTERNALLY to NOT IDENTIFIED or from NOT IDENTIFIED to IDENTIFIED EXTERNALLY.

The IDENTIFIED EXERNALLY clause indicates that SQL inherits the roles defined by the facilities of the operating system, such as rights identifiers.

The NOT IDENTIFIED clause indicates that SQL does not inherit any roles defined by the facilities of the operating system; instead, the role is private to the database. This is the default.

RENAME TO new-role-name

Changes an existing role name to a new role name without changing the privileges granted to the role. You might change the name of a role that corresponds to a department name when the department is renamed. For example, if the personnel department is renamed human resources, you might change the role used by that department from PERSONNEL to HUMAN_RESOURCES. The new role name must not already exist in the database. The old role name is removed from the database when the transaction is committed. The old role name can be re-created and reused, if desired. If the new role name is identified externally, then it must exist as an operating system group or rights identifier.

COMMENT IS ’string’

Adds a comment about the role. SQL displays the text of the comment when it executes a SHOW ROLES statement. Enclose the comment in single quotation marks (’) and separate multiple lines in a comment with a slash mark (/).

Usage Notes

- You must have the SECURITY privilege on the database to alter a role.
- The SHOW PROTECTION and SHOW PRIVILEGE statements will display the new role name created by the ALTER ROLE statement.
- Oracle Rdb rightcases all user and role names that are identified externally. **Rightcasing** means that the names are stored as they are specified in the operating system registry or authorization database. On OpenVMS, all names are converted to uppercase, even if you use delimiters.
- If you issue the RENAME clause, the new role name must exist at the OpenVMS system level.
- If you alter a role using the SQL ALTER ROLE statement and specify the RENAME clause, the new role must exist at the OpenVMS system level.

Examples

Example 1 Renaming a Role

```
SQL> -- Change the name of the role from WRITER to DOCUMENTATION.
SQL> -- Any privileges granted to the role WRITER are transferred to the role
SQL> -- DOCUMENTATION. The role WRITER is deleted from the database.
SQL> ALTER ROLE WRITER
cont> RENAME TO DOCUMENTATION;
SQL> SHOW ROLES;
Roles in database with filename mf_personnel.rdb
     DOCUMENTATION
```

ALTER SEQUENCE Statement

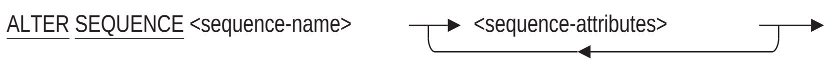
Alters a sequence. A sequence is a database object from which multiple users can generate unique integers. You can use sequences to automatically generate primary key values.

Environment

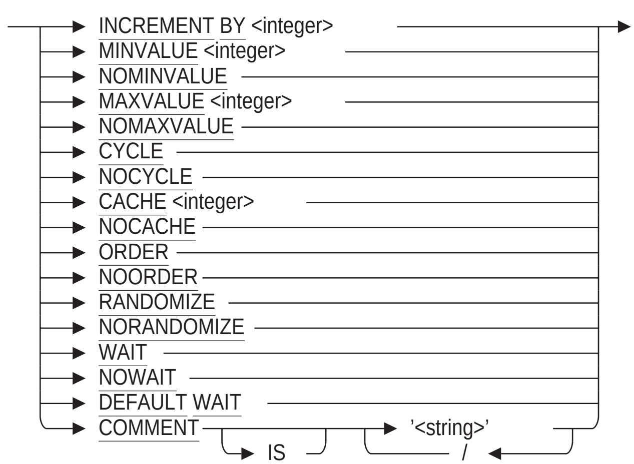
You can use the ALTER SEQUENCE statement:

- In interactive SQL
- Embedded in host language programs to be precompiled
- As part of a procedure in an SQL module
- In dynamic SQL as a statement to be dynamically executed

Format



sequence-attributes =



Arguments

sequence-name
The name of the sequence whose definition you want to change.

INCREMENT BY integer
Specifies the size of the increment and the direction (ascending or descending) of the sequence. This signed quadword (BIGINT) value must be in the range -2147483648 through 2147483647, excluding 0. The absolute value of the integer must be less than the difference between MAXVALUE and MINVALUE. A negative value specifies a descending sequence; a positive value specifies an ascending sequence. If the existing value is positive, then the new value must also be positive. Likewise, if the existing value is negative, then the new value must also be negative. That is, you cannot change a sequence from ascending to descending or from descending to ascending.

By default, the integer value is 1.

MINVALUE integer
NOMINVALUE
The MINVALUE clause specifies the minimum signed quadword (BIGINT) value that the sequence can generate. For an ascending sequence, the new minimum value must be less than or equal to the existing MINVALUE. For a descending sequence, the new minimum value must be less than or equal to the existing RDB$NEXT_SEQUENCE_VALUE. This prevents the minimum value from being greater than any currently issued values. In addition, the integer value must be equal to or greater than -9223372036854775808. The MINVALUE must be less than the integer specified with the MAXVALUE clause. The NOMINVALUE clause specifies that the minimum value for an ascending sequence is 1, and -9223372036854775808 (plus the cache size) for a descending sequence.

MAXVALUE integer
NOMAXVALUE
The MAXVALUE clause specifies the maximum signed quadword (BIGINT) value that the sequence can generate. For an ascending sequence, the new maximum value must be greater than or equal to the existing RDB$NEXT_SEQUENCE_VALUE. For a descending sequence, the new maximum value must be greater than or equal to the existing MAXVALUE. This ensures that the MAXVALUE is not less than any currently issued values. In addition, the integer value must be between -9223372036854775808 and 9223372036854775808. The MAXVALUE must be greater than the integer specified with the MINVALUE clause. The NOMAXVALUE clause specifies that the maximum value for an ascending sequence is

9223372036854775808 (minus the cache size), and -1 for a descending sequence. The NOMAXVALUE clause is the default.

CYCLE
NOCYCLE
The CYCLE clause specifies that the sequence is to continue generating values after reaching either the MINVALUE or MAXVALUE. After an ascending sequence reaches the MAXVALUE, the sequence starts again from its MINVALUE. After a descending sequence reaches its MINVALUE, the sequence starts again at its MAXVALUE. The NOCYCLE clause specifies that the sequence should not continue generating values after reaching either its minimum or maximum value. An error is generated if an attempt is made to increment the sequence beyond its limits.

Note that even after you alter the CYCLE clause, those who were using the sequence at the time you altered the CYCLE clause will continue to use the original setting.

The NOCYCLE clause is the default.

CACHE integer
NOCACHE
The CACHE clause specifies how many values of the sequence Oracle Rdb should preallocate and keep in memory for faster access. The integer value must be a value between 2 and 2147483647. You cannot cache more values than will fit in a given cycle of sequence numbers; thus, the maximum value allowed for the CACHE clause must be less than the value resulting from the following formula:

```
(MAXVALUE-MINVALUE)/ABS(INCREMENT)
```

You can alter the CACHE integer if it is currently a value of 2 or higher. When you alter the CACHE integer, existing users of the sequence continue to use the original setting. You can use the SET FLAGS 'SEQ_CACHE' statement to adjust the cache size for a single process. See the SET FLAGS Statement for details.

If NOCACHE is currently enabled or the CACHE integer is 1, you can alter the CACHE integer, but may have to wait until other users of the sequence have released locks on it. (Note that CACHE 1 is equivalent to NOCACHE.) See the Usage Notes for details.

A cache for a given sequence is populated at the first request for a number from that sequence. If a system failure occurs, all cached sequence values that have not been used in committed SQL statements are lost. The maximum number of lost values is equal to the integer value specified with the CACHE clause. The NOCACHE clause specifies that values of the sequence should not be preallocated.

Note that even after you alter the CACHE value, users who were using the sequence at the time you altered the CACHE will continue to use the original setting.

By default, Oracle Rdb caches 20 sequence values.

ORDER
NOORDER

The ORDER clause specifies that sequence numbers are guaranteed to be assigned in order for each requesting process, thus maintaining a strict history of requests. The NOORDER clause specifies that sequence numbers are not guaranteed to be generated in order of request.

The NOORDER clause is the default.

RANDOMIZE
NORANDOMIZE

The RANDOMIZE clause specifies that the sequence numbers are to be returned with a random value in the most significant bytes of the BIGINT value. This allows unique values to be generated that have a random distribution. When you specify the NORANDOMIZE clause, sequence numbers are close in value to others created at the same time.

The advantage of the RANDOMIZE clause is that updates to columns of a stored index to which these values are written occur in different locations in the index structure and so may improve concurrent access for large indexes that allow leaf nodes in different parts of the index to be updated independently. In contrast, the sequence numbers generated when you specify the NORANDOMIZE clause are likely to be close in numeric value to other sequence values generated at the same time. This may cause index updates to occur in the same or nearby index nodes, which may lead to contention in one part of the sorted index.

The full range of values in the BIGINT value returned for the sequence are used; therefore, the NOMAXVALUE and NOMINVALUE clauses must be specified (or defaulted to) for the sequence definition. The most significant bits of the BIGINT value are set to a randomly generated positive value. A generated distinct value is returned in the least significant 32 bits so that uniqueness is guaranteed. If you also specify the CYCLE clause, then only the least significant 32 bits are cycled. When a query is performed on the column RDB$NEXT_SEQUENCE_VALUE in the RDB$SEQUENCES table, only the generated value of the least significant bits is returned, because the most significant bits are not assigned until the NEXTVAL pseudocolumn is referenced.

If you specify RANDOMIZE, you cannot specify ORDER, MAXVALUE, or MINVALUE. The NORANDOMIZE clause is the default.

WAIT
NOWAIT
DEFAULT WAIT
Specifies what wait state is used when a reference to NEXTVAL is used. A reference to NEXTVAL for a sequence may require synchronization with other users of the sequence. By default (or when you specify DEFAULT WAIT), the wait state (WAIT or NOWAIT) of the current transaction is used. This may mean that no waiting is performed during a NOWAIT transaction.

If you specify WAIT for the sequence, then regardless of the wait state set for the current transaction, all synchronization waits for the next value. This is the recommended setting if the application uses NOWAIT transactions. The current WAIT timeout interval defined for the transaction or database is used.

If you specify NOWAIT for the sequence, then regardless of the current transaction setting, all synchronization will not wait for the next value.

Note that even after you alter the WAIT value, users who were using the sequence at the time you altered WAIT will continue to use the original setting.

COMMENT IS 'string'
Adds a comment about the sequence. SQL displays the text of the comment when it executes a SHOW SEQUENCE statement. Enclose the comment in single quotation marks (') and separate multiple lines in a comment with a slash mark (/).

Usage Notes

- You must have the ALTER privilege on the sequence to alter a sequence.
- The START WITH value cannot be altered. Once a sequence is created, the initial value is established. Drop and then re-create the sequence if the existing starting value is not acceptable.
- If another user holds an exclusive lock on a sequence when you attempt to alter the sequence, your process will wait to execute the statement until the other user commits or rolls back his or her transaction. An exclusive lock is placed on a sequence when a user is altering any of the following attributes:
  - INCREMENT by integer
  - MINVALUE to NOMINVALUE, or the reverse
  - MAXVALUE to NOMAXVALUE, or the reverse

  - NOCACHE to CACHE
  - ORDER to NOORDER, or the reverse
- The value for the RDB$LAST_ALTERED column in the RDB$SEQUENCES system relation is updated by the ALTER SEQUENCE command. The value of RDB$NEXT_SEQUENCE_VALUE is not altered by this command.

Examples

Example 1 Altering a Sequence

```
SQL> -- Show current sequence definition:
SQL> --
SQL> SHOW SEQUENCE EMPIDS
     EMPIDS
 Sequence Id: 1
 Initial Value: 1
 Minimum Value: 1
 Maximum Value: 9223372036854775787
 Next Sequence Value: 1
 Increment by: 1
 Cache Size: 20
 No Order
 No Cycle
 No Randomize
SQL> --
SQL> -- Alter the sequence.
SQL> --
SQL> ALTER SEQUENCE EMPIDS
cont> MINVALUE 0
cont> MAXVALUE 2000
cont> CACHE 30
cont> ORDER
cont> CYCLE;
SQL> --
SQL> -- Show new definition.
SQL> --
SQL> SHOW SEQUENCE EMPIDS
     EMPIDS
 Sequence Id: 1
 Initial Value: 1
 Minimum Value: (none)
 Maximum Value: 2000
 Next Sequence Value: 1
```

```
Increment by: 1
Cache Size: 30
Order
Cycle
No Randomize
```

ALTER STORAGE MAP Statement

Changes an existing storage map. A storage map controls which rows of a table are stored in which storage areas in a multifile database.

In addition to changing storage maps, the ALTER STORAGE MAP statement has options that change the following:

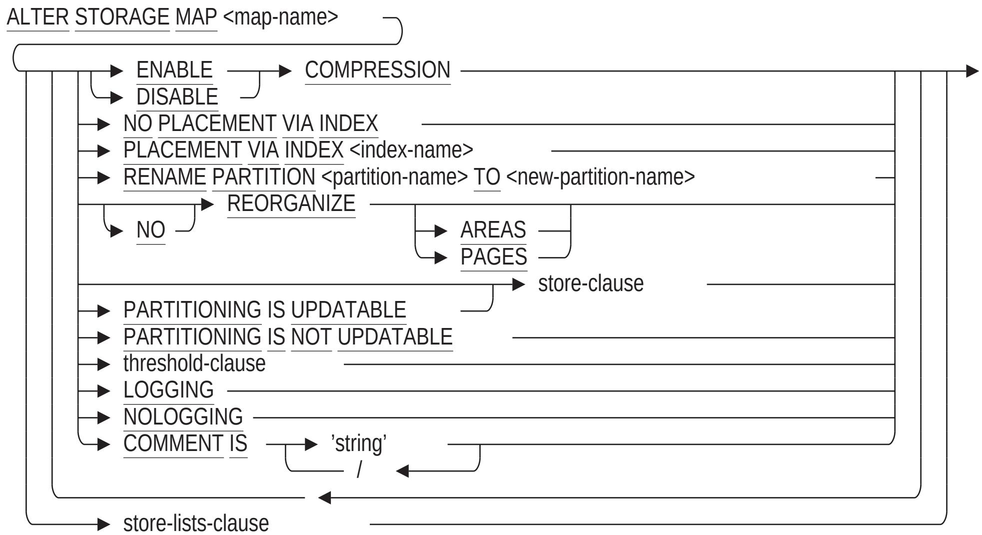
- Which index the database system uses when inserting rows in the table
- Whether or not the rows of the table are stored in a compressed format
- Whether or not the data is reorganized
- Whether partitioning keys can be modified
- Whether logging the transaction containing the ALTER statement is journaled to the .ruj and .aij files.

Environment

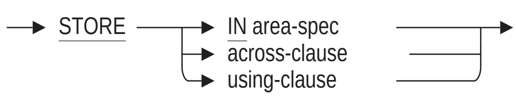
You can use the ALTER STORAGE MAP statement:

- In interactive SQL
- Embedded in host language programs to be precompiled
- As part of a procedure in an SQL module
- In dynamic SQL as a statement to be dynamically executed

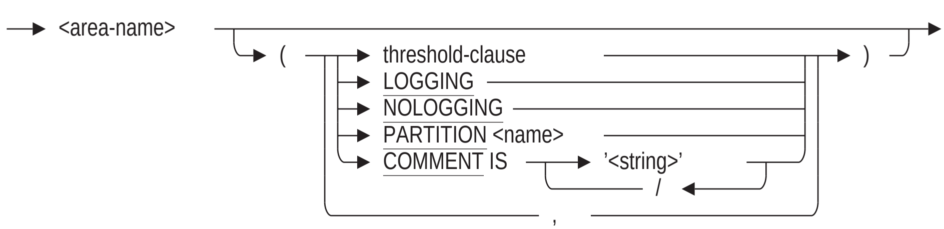
Format



store-clause =

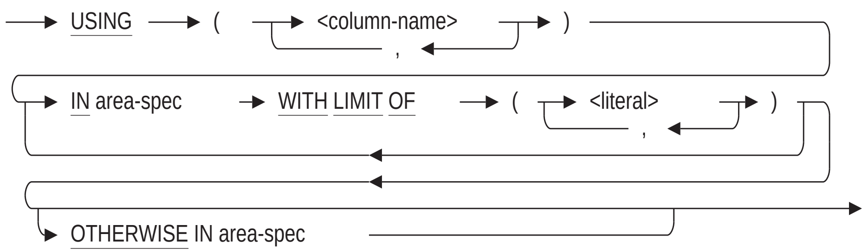


area-spec =

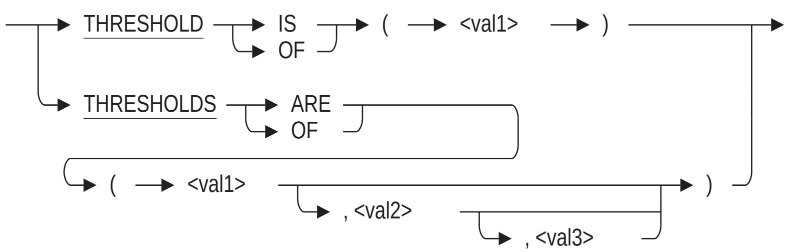


across-clause =

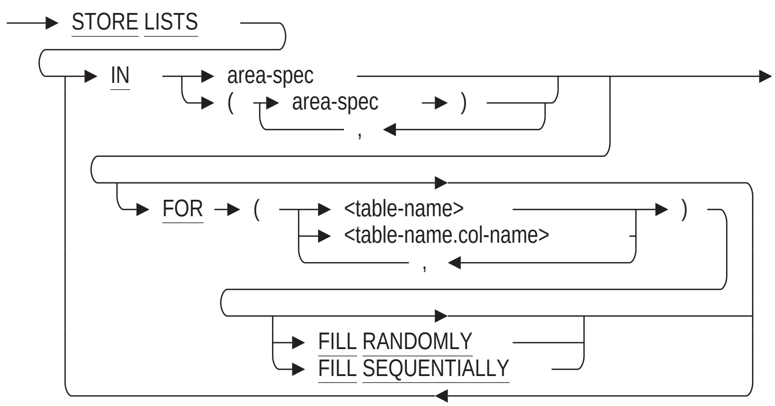
using-clause =



threshold-clause =



store-lists-clause =



Arguments

LOGGING
NOLOGGING
The LOGGING clause specifies that the ALTER STORAGE MAP statement should be logged in the recovery-unit journal file (.ruj) and after-image journal file (.aij).

The NOLOGGING clause specifies that the ALTER STORAGE MAP statement should not be logged in the recovery-unit journal file (.ruj) and after-image journal file (.aij).

The LOGGING clause is the default.

See Section 2.2.8 for information on the advantages and disadvantages of specifying these clauses and the implications that using the NOLOGGING keyword has on database recovery.

PARTITION name
Names the partition. The name can be a delimited identifier if the quoting rules are set to SQL92 or SQL99. Partition names must be unique within the storage map. If you do not specify this clause, Oracle Rdb generates a default name for the partition.

COMMENT IS 'string'

Adds or alters a comment about the storage map. SQL displays the text of the comment when it executes a SHOW STORAGE MAPS statement. Enclose the comment in single quotation marks (') and separate multiple lines in a comment with a slash mark (/).

Usage Notes

- See Section 2.2.8 for information on how disabling logging interacts with other database operations.
- An existing storage map can be converted to a strictly partitioned storage map using the ALTER STORAGE MAP ... PARTITIONING IS NOT UPDATABLE clause.

  This statement implicitly performs a reorganize operation on the base table, moving rows within the map if necessary, but at least scanning the storage areas to make sure all the stored data conforms to the storage map definition. This allows the Oracle Rdb optimizer to use this type of table efficiently when a sequential scan uses a subset of the storage areas.

  In many cases, the database administrator knows that a large table is already strictly partitioned, but it is prohibitive to reorganize the table. The amount of I/O alone might last several hours. Therefore, the database administrator can bypass the automatic reorganize operation performed by the ALTER STORAGE MAP ... PARTITIONING IS NOT UPDATABLE clause by using a NO REORGANIZE clause.

  Because Oracle Rdb has not validated the table partitioning, there is a risk that rows may be missed by sequential scans. The database administrator must take this risk into account when using this clause. Oracle Corporation suggests that an ALTER STORAGE MAP ... REORGANIZE operation be carried out as soon as practical.

  When the NO REORGANIZE clause is used, Oracle Rdb records this information in the Oracle Rdb system relations. The SHOW STORAGE MAP statement will display informational text.
- The NO REORGANIZE clause is ignored unless used with PARTITIONING IS NOT UPDATABLE. This is because either no automatic reorganization is required, or a full rebuild of the table is needed to implement the new map structure.
- The REORGANIZE and NO REORGANIZE clause may not appear in the same ALTER STORAGE MAP command, as shown in the following example:

```
SQL> ALTER STORAGE MAP EMPLOYEES_MAP
cont>     PARTITIONING IS NOT UPDATABLE
cont>     NO REORGANIZE
cont>     REORGANIZE AREAS
cont>     STORE
cont>         USING (EMPLOYEE_ID)
cont>             IN EMPIDS_LOW
cont>                 WITH LIMIT OF ('00200')
cont>             IN EMPIDS_MID
cont>                 WITH LIMIT OF ('00400')
cont>             OTHERWISE IN EMPIDS_OVER;
%SQL-F-MULTSPECATR, Multiple specified attribute. "REORGANIZE" was specified
more than once
```

- The SET FLAGS option, STOMAP_STATS, will output an indication that NO REORGANIZE was used.
- The SHOW STORAGE MAPS statement will output an indication that NO REORGANIZE was used, as shown in the following example:

```
SQL> SHOW STORAGE MAPS EMPLOYEES_MAP
     EMPLOYEES_MAP
 For Table:             EMPLOYEES
 Placement Via Index:   EMPLOYEES_HASH
 Partitioning is:       NOT UPDATABLE
 Strict partitioning was not validated for this table
 ...
```

Examples

Example 1 Disabling Logging to the RUJ and AIJ files

```
SQL> ATTACH 'FILENAME MF_PERSONNEL.RDB';
SQL> ALTER STORAGE MAP EMPLOYEES_MAP
cont>   STORE
cont>       USING (EMPLOYEE_ID)
cont>           IN EMPIDS_LOW
cont>               WITH LIMIT OF ('00200')
cont>           IN JOBS
cont>               (NOLOGGING)
cont>               WITH LIMIT OF ('00400')
cont>           OTHERWISE IN EMPIDS_OVER;
%RDB-W-META_WARN, metadata successfully updated with the reported warning
-RDMS-W-DATACMIT, unjournaled changes made; database may not be recoverable
```

Example 2 Disabled Area Scan for PARTITIONING IS NOT UPDATABLE

```
SQL> SET FLAGS 'stomap_stats';
SQL> ALTER STORAGE MAP EMPLOYEES_MAP
cont>     PARTITIONING IS NOT UPDATABLE
cont>     NO REORGANIZE
cont>     STORE
cont>         USING (EMPLOYEE_ID)
cont>             IN EMPIDS_LOW
cont>                 WITH LIMIT OF ('00200')
cont>             IN EMPIDS_MID
cont>                 WITH LIMIT OF ('00400')
cont>             OTHERWISE IN EMPIDS_OVER;
~As: starting map restructure...
~As: REORGANIZE needed to preserve strict partitioning
~As: NO REORGANIZE was used to override scan
~As: reads: async 0 synch 21, writes: async 7 synch 3
SQL>
SQL> SHOW STORAGE MAPS EMPLOYEES_MAP
     EMPLOYEES_MAP
 For Table:             EMPLOYEES
 Placement Via Index:   EMPLOYEES_HASH
 Partitioning is:       NOT UPDATABLE
 Strict partitioning was not validated for this table
 Comment:        employees partitioned by "00200" "00400"
 Store clause:          STORE
             using (EMPLOYEE_ID)
                 in EMPIDS_LOW
                     with limit of ('00200')
                 in EMPIDS_MID
                     with limit of ('00400')
                 otherwise in EMPIDS_OVER
 Compression is:        ENABLED
SQL>
```

A subsequent ALTER STORAGE MAP ... REORGANIZE statement will validate the partitioning, as shown in the following example:

```
SQL> ALTER STORAGE MAP EMPLOYEES_MAP
cont>     PARTITIONING IS NOT UPDATABLE
cont>     REORGANIZE;
~As: starting map restructure...
~As: starting REORGANIZE...
~As: reorganize AREAS...
~As: processing rows from area 69
```

```
~As: processing rows from area 70
~As: processing rows from area 71
~As: reads: async 408 synch 22, writes: async 3 synch 0
SQL>
```

ALTER SYNONYM Statement

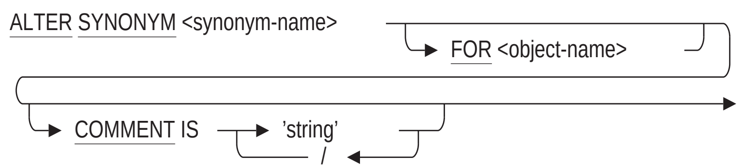
Alters a synonym definition.

Environment

You can use the ALTER SYNONYM statement:

- In interactive SQL
- Embedded in host language programs
- As part of a procedure in an SQL module
- In dynamic SQL as a statement to be dynamically executed

Format



Arguments

synonym-name
The name of an existing synonym you want to alter.

FOR object-name
You may change the synonym to reference a different database object; however, it must be of the same type. Oracle Rdb assumes that the object has the same or similar characteristics as the referenced object. The referenced object must exist in the database.

COMMENT IS 'string'
This clause can be used to add several lines of comment to the synonym object. The SHOW SYNONYM statement displays the comment.

Usage Notes

- An error is generated if this statement is used on a database that has not been enabled for synonyms. See the ALTER DATABASE ... SYNONYMS ARE ENABLED clause.
- You must have database ALTER privilege in order to execute the ALTER SYNONYM statement.
- You must have REFERENCES privilege on the referenced object to alter a synonym for that object. Because domains do not have access control, no other privileges are required to alter synonyms for domains.
- You can alter synonyms for synonyms. Therefore, it is possible to create a cycle within a chain of synonyms. Oracle Rdb will detect this cycle and reject the definition.

Examples

Example 1 Adding a Comment

```
SQL> ALTEr SYNONYM CASH
cont>   COMMENT IS 'use a different name to avoid confusion with'
cont>   /          'the domain MONEY';
```

Example 2 Using Multiple Synonyms and Changing the Referenced Table Using ALTER

```
SQL> CREATE TABLE t_employees_0001 (...);
SQL> CREATE SYNONYM employees FOR t_employees_0001;
SQL> CREATE SYNONYM emps FOR employees;
SQL> CREATE TABLE t_employees_0002 (...);
SQL> ALTER SYNONYM employees FOR t_employees_0002;
```

ALTER TABLE Statement

Changes an existing table definition. You can:

- Add columns
- Add constraints to tables or columns
- Enable constraints
- Modify columns
- Modify character sets
- Modify data types
- Delete columns
- Disable constraints
- Delete constraints

The ALTER TABLE statement can also add or delete table-specific constraints, updating the physical database appropriately. These constraints can be deleted, declared, or both. You cannot alter an existing constraint; instead, you must specifically delete it by name and then create it again with the definition you want. You can display the names for all constraints currently associated with a table by using the SHOW TABLE statement. Any number of constraints can be deleted and declared at both the table and column levels.

When you execute this statement, SQL modifies the named column definitions in the table definition. All of the columns that you do not mention remain the same. SQL defines new versions of columns *before* defining constraints. Then, SQL defines and evaluates constraints before storing them. Therefore, if columns and constraints are defined in the same table definition, constraints always apply to the latest version of a column.

When you change a table definition, other users see the revised definition only when they declare the schema after you commit the changes.

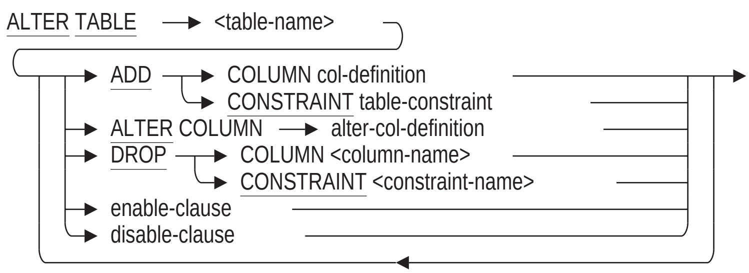
Environment

You can use the ALTER TABLE statement:

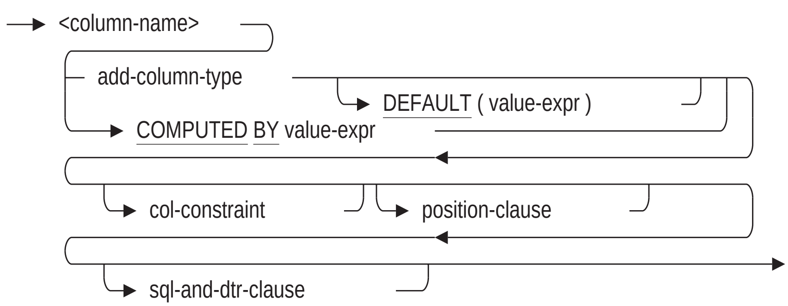
- In interactive SQL

- Embedded in host language programs to be precompiled
- As part of a procedure in an SQL module
- In dynamic SQL as a statement to be dynamically executed

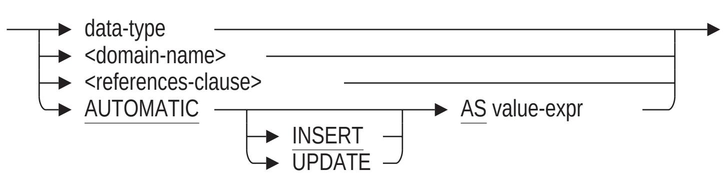
Format



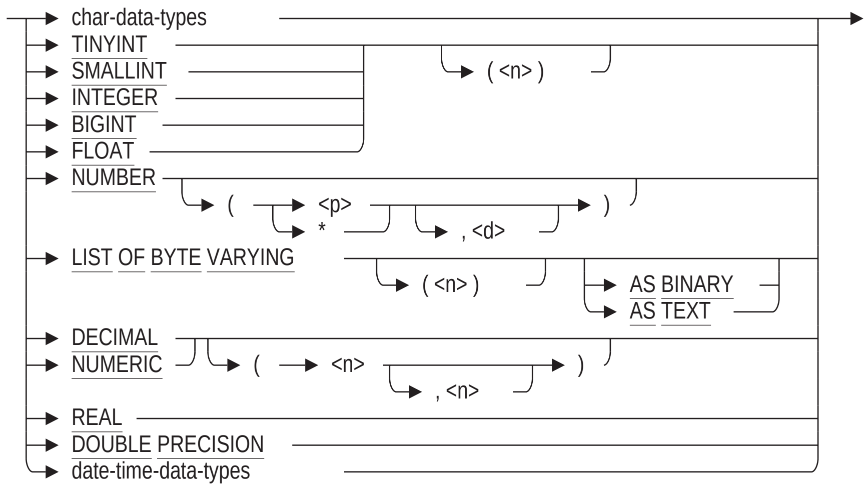
col-definition =



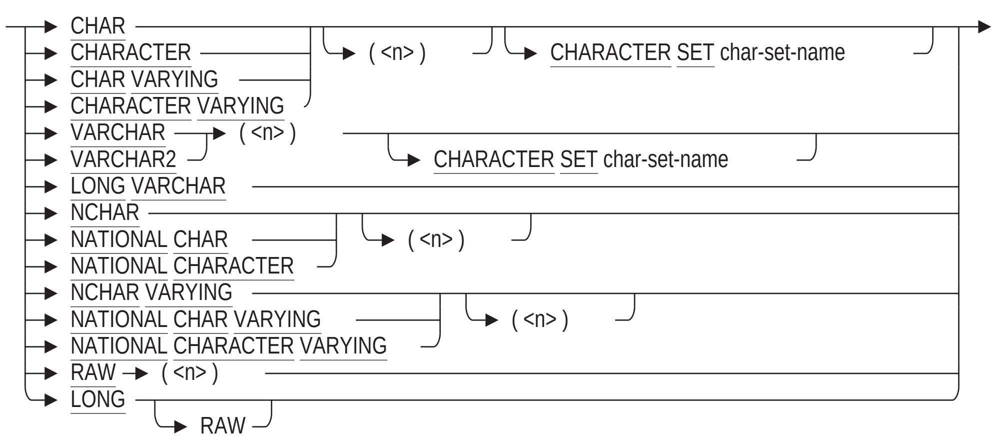
add-column-type =



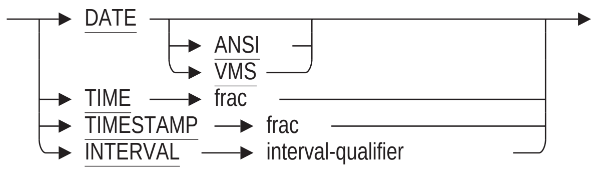
data-type =



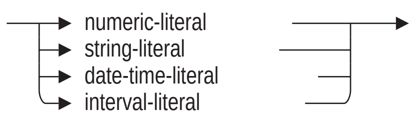
char-data-types =



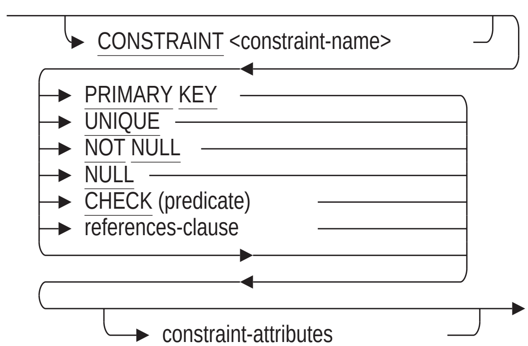
date-time-data-types =



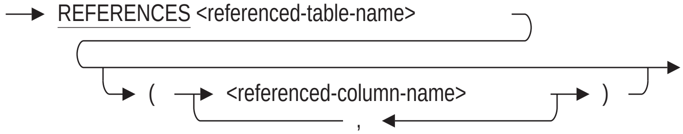
literal =



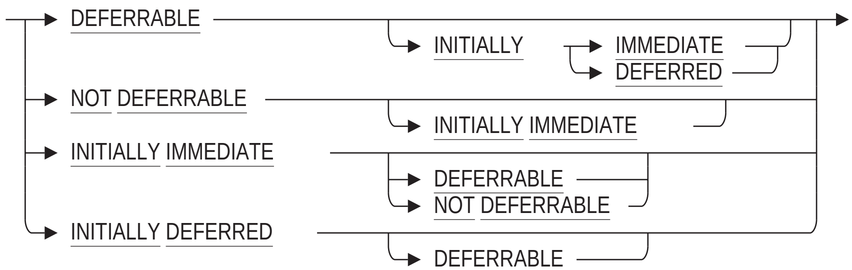
col-constraint=



references-clause =



constraint-attributes =



position-clause =

sql-and-dtr-clause =

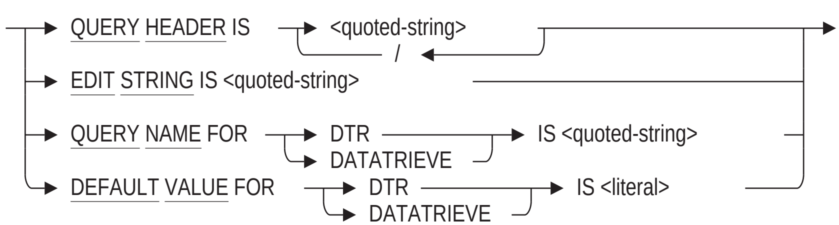


table-constraint =

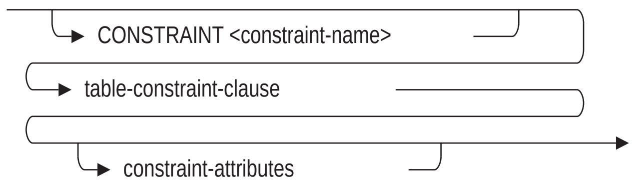
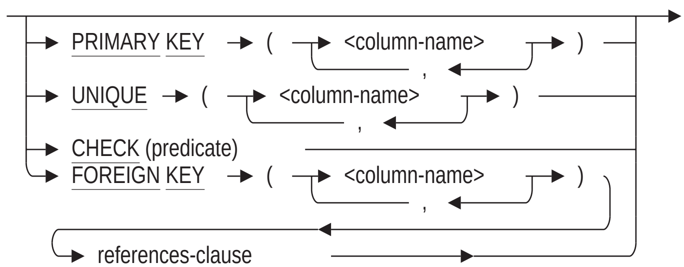
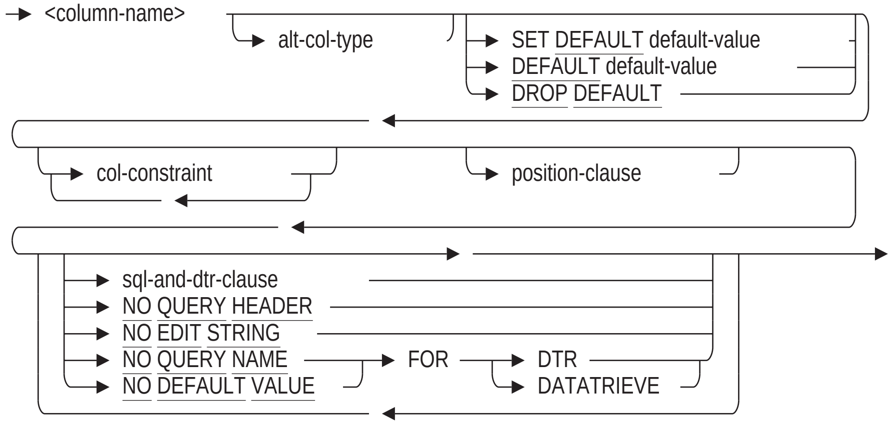


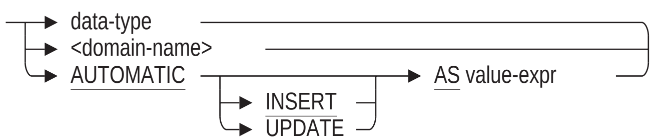
table-constraint-clause =



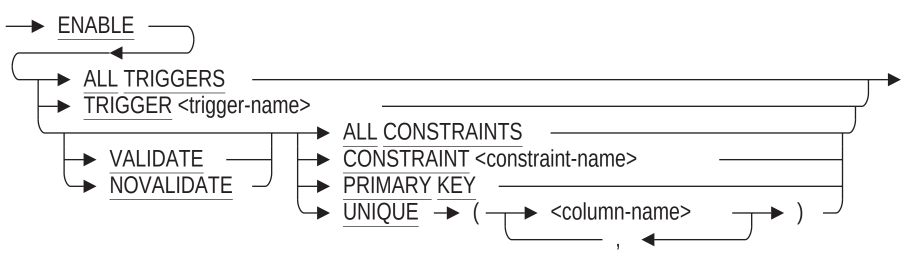
alter-col-definition =



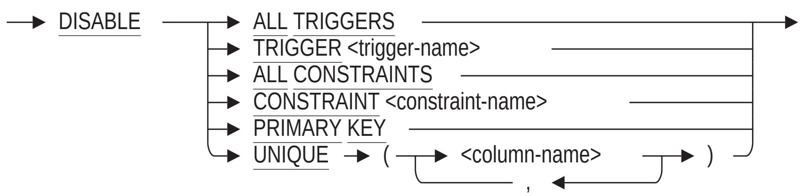
alt-col-type =



enable-clause =



disable-clause =



Arguments

NULL

Specifies that NULL is permitted for the column. This is the default behavior. A column with a NULL constraint cannot also have a NOT NULL constraint within the same ALTER TABLE statement. However, no checks are performed for CHECK constraints, which may limit the column to non-null values.

The NULL constraint is not stored in the database and is provided only as a syntactic alternative to NOT NULL.

Because the NOT NULL constraint specifies that all values except NULL can be inserted, it might seem logical to think that a NULL constraint specifies that only NULL values can be inserted. However, this is not true. A NULL constraint specifies that both NULL and non-null values are allowed.

references-clause

When used as the column type clause, specifies that the type of the column be inherited from the PRIMARY KEY or UNIQUE index referenced. Both the data type and domain are inherited.

AUTOMATIC AS value-expr
AUTOMATIC INSERT AS value-expr
AUTOMATIC UPDATE AS value-expr

These AUTOMATIC column clauses allow you to store special information when data is inserted into a row or a row is updated. For example, you can log application-specific information to audit activity or provide essential values, such as time stamps or unique identifiers for the data.

The assignment of values to these types of columns is managed by Oracle Rdb. The AUTOMATIC INSERT clause can be used to provide a complex default for the column when the row is inserted; it cannot be changed by an UPDATE statement.

The AUTOMATIC UPDATE clause can be used to provide an updated value during an UPDATE statement. The AUTOMATIC clause is the default and specifies that the value expression should be applied during both INSERT and UPDATE statements. The column type is derived from the AS value-expr; using CAST allows a specific data type to be specified. However, this is not required and is rarely necessary.

You can define an AUTOMATIC INSERT column to automatically receive data during an insert operation. The data is stored like any other column, but the column is read-only. Because AUTOMATIC columns are treated as read-only columns, they cannot appear in the column list for an insert operation nor be modified by an update operation. AUTOMATIC UPDATE columns can have an associated default value that will be used when the row is inserted.

Suppose that you want to store the current time stamp of a transaction and supply a unique numeric value for an order number. In addition, when the row is updated (the order is altered), you want a new time stamp to be written to the LAST_UPDATED column. You could write an application to supply this information, but you could not guarantee the desired behavior. For instance, a user with access to the table might update the table with interactive SQL and forget to enter a new time stamp to the LAST_UPDATED column. If you use an AUTOMATIC column instead, it can be defined so that columns automatically receive data during an insert operation. The data is sorted like any other column, but the column is read-only.

See the Usage Notes for more information on automatic columns.

DEFAULT value-expr

Provides a default value for a column if the row that is inserted does not include a value for that column.

You can add a default value to an existing column or alter the existing default value of a column by altering the table. However, doing so has no effect on the values stored in existing rows.

If you do not specify a default value, a column inherits the default value from the domain. If you do not specify a default value for either the column or domain, SQL assigns NULL as the default value.

If you specify a default value for either the column or domain when a column is added, SQL propagates the default value from the column or domain to all previously stored rows. Therefore, when you add a column to a table and specify a default value for the column, SQL stores the default value in the newly added column of all previously stored rows. Likewise, if the newly added column is based upon a domain that specifies a default value, SQL stores the default value in the column of all previously stored rows. See Example 3 in the Examples section.

Because SQL updates data when you add a column with a default value other than NULL, the ALTER TABLE statement can take some time to complete when the table contains many rows. (If you specify a default value of NULL, SQL does not modify the data because SQL automatically returns a null value for columns that have no actual value stored in them.) If you want to add more than one column with default values, add them in one ALTER TABLE statement. When you do so, SQL scans the table data once instead of many times.

Because data is added to the rows, adding a column with a default value may result in fragmented records. For information about locating and correcting record fragmentation, see the *Oracle Rdb7 Guide to Database Performance and Tuning.*

Remember that the default value for a column is not the same as the missing value that you can specify using the RDO interface. See the *Oracle Rdb7 Guide to Database Design and Definition* for information on the difference between a default value and a missing value.

default-value
Specifies the default value of a column.

constraint-attributes
Although the constraint attribute syntax, shown in Table 3–4, provides 11 permutations as required by the SQL92 or SQL99 standard, they equate to the following three options:

- INITIALLY IMMEDIATE NOT DEFERRABLE

  Specifies that evaluation of the constraint must take place when the INSERT, DELETE, or UPDATE statement executes. If you are using the SQL92, SQL99, MIA, or ORACLE LEVEL1 dialect, this is the default.

  This clause is the same as the NOT DEFERRABLE option provided in previous releases of Oracle Rdb.

- INITIALLY DEFERRED DEFERRABLE

  Specifies that evaluation of the constraint can take place at any later time. Unless otherwise specified, evaluation of the constraint takes place as the COMMIT statement executes. You can use the SET ALL CONSTRAINTS statement to have all constraints evaluated earlier. See the description of the SET ALL CONSTRAINTS statement in the *Oracle Rdb7 SQL Reference Manual* for more information. If you are using the default SQLV40 dialect, this is the default constraint attribute. When using this default dialect, Oracle Rdb displays a deprecated feature message for all constraints defined without specification of one of the constraint attributes.

This clause is the same as the DEFERRABLE option provided in previous releases of Oracle Rdb.

- INITIALLY IMMEDIATE DEFERRABLE

  Specifies that evaluation of the constraint be deferred (using the SET ALL CONSTRAINTS statement or the SET TRANSACTION statement with the EVALUATING clause), but by default it is evaluated after the INSERT, DELETE, or UPDATE statement executes. This option is new for Oracle Rdb release 7.1.

Table 3–4 Constraint Attributes Syntax Permutations and Equivalents

If You Specify This Clause:	It Defaults to This Clause:
Do not specify a clause	INITIALLY IMMEDIATE NOT DEFERRABLE
NOT DEFERRABLE	
INITIALLY IMMEDIATE	
INITIALLY IMMEDIATE NOT DEFERRABLE	
NOT DEFERRABLE INITIALLY IMMEDIATE	
INITIALLY DEFERRED	INITIALLY DEFERRED DEFERRABLE
DEFERRABLE INITIALLY DEFERRED	
INITIALLY DEFERRED DEFERRABLE	
DEFERRABLE	INITIALLY IMMEDIATE DEFERRABLE
INITIALLY IMMEDIATE DEFERRABLE	
DEFERRABLE INITIALLY IMMEDIATE	

AFTER COLUMN column-name
BEFORE COLUMN column-name

Changes the normal field ordering of columns to make the displayed column ordering more readable. Note that this does not change the on-disk layout of the columns. By default, when neither of these clauses is specified, columns are positioned at the end of the table specified with the ALTER TABLE statement.

enable-clause
disable-clause
Allows you to enable or disable all triggers, specified triggers, all constraints, specified constraints, a primary key, or a unique constraint, as described in the following list. By default, table and column constraints added during an alter table operation are enabled.

- DISABLE ALL TRIGGERS

  All triggers defined for the table are disabled. (No error is raised if no triggers are defined for this table.)

- ENABLE ALL TRIGGERS

  All triggers defined for the table are enabled. (No error is raised if no triggers are defined for this table.)

- DISABLE TRIGGER trigger-name

  The named trigger for this table is disabled. The named trigger must be defined on the table.

- ENABLE TRIGGER trigger-name

  The named trigger for this table is enabled. The named trigger must be defined on the table.

- DISABLE ALL CONSTRAINTS

  All table and column constraints for this table are disabled. (No error is raised if no constraints are defined on the table.)

- ENABLE ALL CONSTRAINTS

  All table and column constraints for this table are enabled. (No error is raised if no constraints are defined on the table.)

- DISABLE CONSTRAINT constraint-name

  The named constraint is disabled. The named constraint must be a table or column constraint for the table.

- ENABLE CONSTRAINT constraint-name

  The named constraint is enabled. The named constraint must be a table or column constraint for the table.

- DISABLE PRIMARY KEY

  The primary key for the table is disabled.

- ENABLE PRIMARY KEY

  The primary key for the table is enabled.

- DISABLE UNIQUE (column-name)

  The matching UNIQUE constraint is disabled. The columns listed must be columns in the table.

- ENABLE UNIQUE (column-name)

  The matching UNIQUE constraint is enabled. The columns listed must be columns in the table.

- VALIDATE and NOVALIDATE

  These options are available only on the enable-clause. By default, table and column constraints are enabled during an ALTER TABLE statement. When a constraint is added or enabled with the ALTER TABLE statement, the default is to validate the table contents. The ENABLE NOVALIDATE option allows a knowledgeable database administrator to avoid the time and I/O resources required to revalidate the data when he or she knows the data is valid.

Note: Oracle Corporation recommends that you use the RMU Verify command with the Constraint qualifier periodically to verify that your assumptions are correct if you use the ENABLE NOVALIDATE option.

Usage Notes

- The following usage notes apply to constraints:
  - In addition to the ALTER privilege that is required to issue the ALTER TABLE statement, you must have the DROP privilege on the table to disable a constraint or trigger, and you must have the CREATE privilege on the table to enable a constraint or a trigger. You must hold the DBADM privilege on the database to specify the NOVALIDATE option to the enable-clause.
  - When a constraint is disabled, it is not evaluated by the INSERT, UPDATE, or DELETE statements.
  - When a previously disabled constraint is reenabled, the constraint is validated to ensure that all existing rows are valid.

  - If you enable constraints during an alter table operation, and do not specify NOVALIDATE, each constraint will be validated. If the referential integrity constraint fails, then the alter table operation fails also and the failed constraint name will be reported.
  - The RMU Verify command with the Constraint qualifier ignores any disabled constraint unless that constraint is named with the Constraint qualifier and the Constraint option. If the Constraint qualifier specifies all constraints (no options are specified), or if it specifies a specific table, all disabled constraints are ignored. This allows you to check a disabled constraint periodically without the need to reenable it, which might be useful if the overhead of checking the constraint during operating hours is too expensive, or if it is already being enforced by the application.

- The following usage notes apply to AUTOMATIC columns:
  - When the column is omitted from an insert operation, a column default and an automatic column provide similar functions. However, there are distinctions, as follows:
    - AUTOMATIC columns can execute complex expressions, including select expressions and function calls.
    - AUTOMATIC columns cannot be referenced during an insert operation because they are read-only to applications.
    - AUTOMATIC columns can be active during an update operation.
    - When you use an AUTOMATIC column, you do not provide the data type for the column.
  - Note the following differences between using COMPUTED BY columns and AUTOMATIC columns:
    - COMPUTED BY columns use no space in the row, AUTOMATIC columns use space.
    - A COMPUTED BY column is evaluated when the row is fetched, such as when a SELECT, UPDATE, or DELETE statement references the column name. An AUTOMATIC column is evaluated during an INSERT or UPDATE statement. A calculated value is written to a column in the row, and the value returned by a SELECT statement is the stored column value.

      For example, a column defined as COMPUTED BY CURRENT_DATE returns the date when the query is executed. A selected column that is AUTOMATIC INSERT AS CURRENT_DATE returns the date when the

INSERT was performed, which might be different from the date when the query is executed.

- Note the following differences between using an AUTOMATIC column and a trigger on the table:
  * In an insert operation, an AFTER INSERT TRIGGER trigger can provide AUTOMATIC column functionality. However, AUTOMATIC columns can help eliminate the overhead of a trigger and so simplify table management.
  * Trigger actions cannot modify a row being updated, because this leads to a recursive trigger action. AUTOMATIC UPDATE columns are evaluated prior to the trigger and constraint execution.
- You can convert existing columns to AUTOMATIC. However, this does not affect existing data in the table. It affects only further inserts and updates. Applications that previously wrote to these columns must be modified.
- If the data written to the table with an AUTOMATIC column is incorrect, you can temporarily suspend the read-only attribute of the column by issuing the SET FLAGS 'AUTO_OVERRIDE' statement if you have the DBADMIN privilege on the database. Then you can execute an update query to correct the incorrect data. See the SET FLAGS Statement for more information and an example.
- You can create an index on an AUTOMATIC column. AUTOMATIC columns are identical to other columns, except that Oracle Rdb, not a user application, assigns the value.

Examples

Example 1 Using the Position Clause

```
SQL> SHOW TABLE (COL) EMPLOYEES
Information for table EMPLOYEES
Columns for table EMPLOYEES:
Column Name                     Data Type        Domain
-----------                     ---------        ------
EMPLOYEE_ID                     CHAR(5)          ID_NUMBER
 Missing Value:
LAST_NAME                       CHAR(14)         LAST_NAME
FIRST_NAME                      CHAR(10)         FIRST_NAME
MIDDLE_INITIAL                  CHAR(1)          MIDDLE_INITIAL
 Missing Value:
```

```
ADDRESS_DATA_1                CHAR(25)         ADDRESS_DATA_1
 Missing Value:
ADDRESS_DATA_2                CHAR(25)         ADDRESS_DATA_2
 Missing Value:
CITY                          CHAR(20)         CITY
 Missing Value:
STATE                         CHAR(2)          STATE
 Missing Value:
POSTAL_CODE                   CHAR(5)          POSTAL_CODE
 Missing Value:
SEX                           CHAR(1)          SEX
 Missing Value: ?
BIRTHDAY                      DATE VMS         STANDARD_DATE
 Missing Value: 17-NOV-1858 00:00:00.00
STATUS_CODE                   CHAR(1)          STATUS_CODE
 Missing Value: N
SQL> -- Alter the table to rearrange the order in which columns
SQL> -- are displayed.
SQL> ALTER TABLE EMPLOYEES
cont> ALTER COLUMN SEX BEFORE COLUMN LAST_NAME
cont> ALTER COLUMN BIRTHDAY BEFORE COLUMN LAST_NAME
cont> ALTER COLUMN STATUS_CODE BEFORE COLUMN LAST_NAME;
SQL> COMMIT;
SQL> -- Show the table to demonstrate that the order in which
SQL> -- columns are displayed has changed.
SQL> SHOW TABLE (COL) EMPLOYEES;
Information for table EMPLOYEES
Columns for table EMPLOYEES:
Column Name                   Data Type        Domain
-----------                   ---------        ------
EMPLOYEE_ID                   CHAR(5)          ID_NUMBER
 Missing Value:
SEX                           CHAR(1)          SEX
 Missing Value: ?
BIRTHDAY                      DATE VMS         STANDARD_DATE
 Missing Value: 17-NOV-1858 00:00:00.00
STATUS_CODE                   CHAR(1)          STATUS_CODE
 Missing Value: N
LAST_NAME                     CHAR(14)         LAST_NAME
FIRST_NAME                    CHAR(10)         FIRST_NAME
MIDDLE_INITIAL                CHAR(1)          MIDDLE_INITIAL
 Missing Value:
ADDRESS_DATA_1                CHAR(25)         ADDRESS_DATA_1
 Missing Value:
ADDRESS_DATA_2                CHAR(25)         ADDRESS_DATA_2
```

```
 Missing Value:
CITY                            CHAR(20)         CITY
 Missing Value:
STATE                           CHAR(2)          STATE
 Missing Value:
POSTAL_CODE                     CHAR(5)          POSTAL_CODE
 Missing Value:
```

Example 2 Disabling a Trigger

```
SQL> SELECT * FROM JOB_HISTORY WHERE EMPLOYEE_ID='00164';
 EMPLOYEE_ID   JOB_CODE   JOB_START     JOB_END       DEPARTMENT_CODE
   SUPERVISOR_ID
 00164         DMGR       21-Sep-1981   NULL          MBMN
   00228
 00164         SPGM        5-Jul-1980   20-Sep-1981   MCBM
   00164
2 rows selected
SQL> DELETE FROM EMPLOYEES WHERE EMPLOYEE_ID ='00164';
1 row deleted
SQL> -- Show that the EMPLOYEE_ID_CASCADE_DELETE trigger caused
SQL> -- records in the JOB_HISTORY table to be deleted for the
SQL> -- employee with EMPLOYEE_ID of 00164.
SQL> SELECT * FROM JOB_HISTORY WHERE EMPLOYEE_ID='00164';
0 rows selected
SQL> -- Roll back the delete operation and alter the EMPLOYEES table
SQL> -- to disable the EMPLOYEE_ID_CASCADE_DELETE trigger.
SQL> ROLLBACK;
SQL> ALTER TABLE EMPLOYEES
cont> DISABLE TRIGGER EMPLOYEE_ID_CASCADE_DELETE;
SQL> -- Commit the alter operation and disconnect to ensure that
SQL> -- the next connection will have the trigger disabled.
SQL> COMMIT;
SQL> DISCONNECT DEFAULT;
SQL> ATTACH 'FILENAME MF_PERSONNEL.RDB';
SQL> DELETE FROM EMPLOYEES WHERE EMPLOYEE_ID ='00164';
1 row deleted
SQL> -- Show that with the trigger disabled, a deletion of
SQL> -- employee 00164 from the EMPLOYEES table does not
SQL> -- trigger a deletion for that employee from the
SQL> -- JOB_HISTORY table.
SQL> SELECT * FROM JOB_HISTORY WHERE EMPLOYEE_ID='00164';
EMPLOYEE_ID   JOB_CODE   JOB_START     JOB_END       DEPARTMENT_CODE
   SUPERVISOR_ID
 00164         DMGR       21-Sep-1981   NULL          MBMN
```

```
  00228
 00164         SPGM        5-Jul-1980   20-Sep-1981   MCBM
   00164
2 rows selected
```

Example 3 Adding Columns with Default Values to Tables

```
SQL> -- Add the column PHONE and specify a default value.
SQL> --
SQL> ALTER TABLE EMPLOYEES
cont>     ADD PHONE CHAR(7) DEFAULT 'None';
SQL> --
SQL> -- The result table shows that the rows contain the default value
SQL> -- of the PHONE column.
SQL> --
SQL> SELECT LAST_NAME, PHONE FROM EMPLOYEES;
 LAST_NAME        PHONE
 Toliver          None
 Smith            None
 Dietrich         None
 Kilpatrick       None
.
.
.
SQL>
```

ALTER TRIGGER Statement

Enables or disables an existing trigger. Changes take place after the transaction containing the ALTER TRIGGER statement is committed.

Environment

You can use the ALTER TRIGGER statement:

- In interactive SQL
- Embedded in host language programs to be precompiled
- As part of a procedure in an SQL module
- In dynamic SQL as a statement to be dynamically executed

Format

```
ALTER TRIGGER <trigger-name>  ─┬─> DISABLE ─┬─>
                               └─> ENABLE  ─┘
```

Arguments

trigger-name
The name of an existing trigger.

DISABLE
Disables a previously enabled trigger.

ENABLE
Enables a previously disabled trigger.

Usage Notes

- In addition to the ALTER privilege on the owning table, the user must have DROP privilege on the table to use an ALTER TRIGGER DISABLE statement (if the trigger is currently enabled), and the user must have the CREATE privilege on the table to use an ALTER TRIGGER ENABLE statement (if the trigger is currently disabled).

- By default, a trigger is enabled when it is created.
- When a trigger is disabled, it is not executed by the INSERT, UPDATE, or DELETE statement.
- When a previously disabled trigger is reenabled, it takes effect for new queries only. Existing cursors that have been opened do not see the enabled triggers. Therefore, it is recommended that when you enable or disable a trigger, you commit, disconnect, and reattach so that consistent behavior is seen by all queries.
- When you use the RMU Extract command, the setting (disabled or enabled) for a trigger is extracted as an ALTER TRIGGER statement after the trigger is created.
- Use the SHOW TRIGGERS statement to display the setting (disabled or enabled) for a trigger.

Examples

Example 1 Disabling a Trigger

```
SQL> SELECT * FROM JOB_HISTORY WHERE EMPLOYEE_ID='00164';
 EMPLOYEE_ID   JOB_CODE   JOB_START     JOB_END       DEPARTMENT_CODE
   SUPERVISOR_ID
 00164         DMGR       21-Sep-1981   NULL          MBMN
   00228
 00164         SPGM        5-Jul-1980   20-Sep-1981   MCBM
   00164
2 rows selected
SQL> DELETE FROM EMPLOYEES WHERE EMPLOYEE_ID ='00164';
1 row deleted
SQL> SELECT * FROM JOB_HISTORY WHERE EMPLOYEE_ID='00164';
0 rows selected
SQL> ROLLBACK;
SQL> ALTER TRIGGER EMPLOYEE_ID_CASCADE_DELETE DISABLE;
SQL> COMMIT;
SQL> DISCONNECT DEFAULT;
.
.
.
SQL> DELETE FROM EMPLOYEES WHERE EMPLOYEE_ID ='00164';
1 row deleted
SQL> SELECT * FROM JOB_HISTORY WHERE EMPLOYEE_ID='00164';
EMPLOYEE_ID   JOB_CODE   JOB_START     JOB_END       DEPARTMENT_CODE
   SUPERVISOR_ID
```

```
 00164         DMGR       21-Sep-1981   NULL          MBMN
   00228
 00164         SPGM        5-Jul-1980   20-Sep-1981   MCBM
   00164
2 rows selected
```

This example shows that while the EMPLOYEE_ID_CASCADE_DELETE trigger is enabled, deleting a record from EMPLOYEES causes the corresponding record in JOB_HISTORY to be deleted. After the trigger is disabled, a deletion from EMPLOYEES does not trigger a deletion from the JOB_HISTORY table.

ALTER USER Statement

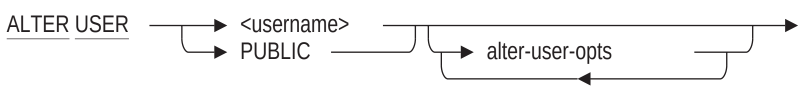
Modifies an entry for the specified user name or user class. The modifications take effect on the next database connection after the ALTER USER statement is committed.

Environment

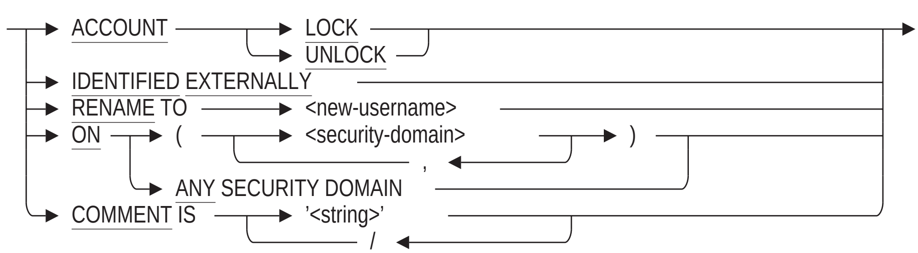
You can use the ALTER USER statement:

- In interactive SQL
- Embedded in host language programs to be precompiled
- As part of a procedure in an SQL module
- In dynamic SQL as a statement to be dynamically executed

Format



alter-user-opts =



Arguments

username

An existing user name in the database. To specify lowercase characters or characters not in the SQL repertoire, enclose the user name in double quotation marks (").

PUBLIC

The PUBLIC user in the database. This entry gives you control over anonymous users who access the database.

ACCOUNT LOCK
ACCOUNT UNLOCK

The ACCOUNT LOCK clause disables access to the database by the user for whom the ALTER USER statement is being applied. The ACCOUNT UNLOCK clause allows the user access to the database.

IDENTIFIED EXTERNALLY

Indicates that the user will be authenticated through the operating system, not using a password. This is the default and the only option currently available.

RENAME TO new-username

Changes the user name and, if a security profile exists, assigns the security profile associated with the old user name to the new user name. This might be used, for example, when a person's name changes (as through marriage), and, therefore, his or her account on the operating system is changed accordingly. The new-username must not currently exist in the database.

When the ALTER USER statement is committed, the existing user name is removed from the database and replaced with the new-username. Subsequent SHOW PROTECTION statements will display the new name for the user, and all GRANT and REVOKE statements will require the new-username. The new-username is not visible to other sessions until the transaction containing the ALTER USER command is committed.

ON (security-domain)
ON ANY SECURITY DOMAIN

The ON security-domain clause specifies one or more security domains from which the user can access the database. If you do not specify an ON clause, then the specified user can access the database from the current security domain only (as determined by the database server). The ON ANY SECURITY DOMAIN clause specifies that this user can access the database from any domain.

COMMENT IS 'string'

Adds a comment about the user. SQL displays the text of the comment when it executes a SHOW USERS statement. Enclose the comment in single quotation marks (') and separate multiple lines in a comment with a slash mark (/).

Usage Notes

- You must have the SECURITY privilege on the database to alter a user.
- You can display existing users defined for a database by issuing a SHOW USERS statement.
- Oracle Rdb rightcases all user and role names that are identified externally. **Rightcasing** means that the names are stored as they are specified in the operating system registry or authorization database. On OpenVMS, all names are converted to uppercase, even if you use delimiters.
- When you issue the RENAME clause, the new user name must exist as a system user name. See the Examples section.

  For example, if you alter a user using the SQL ALTER USER statement and specify the RENAME clause, the new user name must exist at the OpenVMS system level. For example, if user pjones changes her name to pkennedy, pkennedy must be a valid OpenVMS user name.

Examples

Example 1 Renaming a User

```
SQL> ALTER USER "pjones"
cont> RENAME to "pkennedy"
cont> COMMENT IS 'P. Jones married and changed her name to P. Kennedy';
```

CASE (Searched) Control Statement

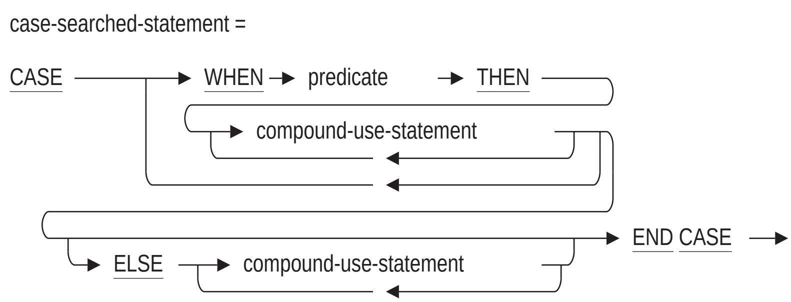
Executes one of a sequence of alternate statement blocks in a compound statement of a multistatement procedure. Unlike the simple CASE control statement, the searched CASE control statement supports arbitrary predicates for the WHEN clause that can contain variable and parameter references.

Environment

You can use the searched CASE control statement in a compound statement of a multistatement procedure:

- In interactive SQL
- Embedded in host language programs to be precompiled
- As part of a procedure in an SQL module
- In dynamic SQL as a statement to be dynamically executed

Format



Arguments

WHEN predicate

Determines whether the compound use statements in the THEN clause are to be executed or the compound use statements in the ELSE clause are to be executed. If the predicate evaluates to TRUE, then the compound use statements in the THEN clause are executed. If the predicate evaluates to FALSE or UNKNOWN, then the compound use statements in the ELSE clause are executed.

THEN compound-use-statement
Specifies the set of SQL statements to be executed when the WHEN clause evaluates to TRUE.

ELSE compound-use statement
Specifies the set of SQL statements to be executed when the WHEN clause evaluates to FALSE or UNKNOWN.

Usage Notes

No new usage notes.

Examples

Example 1 Specifying Predicates with Variable References

```
SQL> CREATE TABLE T (C INT);
SQL> BEGIN
cont> DECLARE :V INTEGER = 10;
cont> DECLARE :X INTEGER = 0;
cont> CASE
cont>   WHEN :V = 1 THEN INSERT INTO T(C) VALUES (:X + 1);
cont>   WHEN :V = 2 THEN INSERT INTO T(C) VALUES (:X + 2);
cont>   WHEN :V > 3 THEN INSERT INTO T(C) VALUES (:X);
cont>   ELSE INSERT INTO T(C) VALUES (-1);
cont> END CASE;
cont> END;
```

CASE (Simple) Control Statement

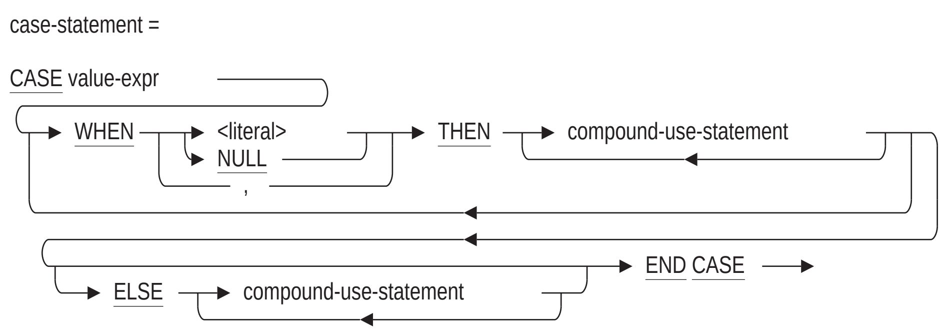
Executes one of a sequence of alternate statement blocks in a compound statement of a multistatement procedure.

Environment

You can use the simple CASE control statement in a compound statement of a multistatement procedure:

- In interactive SQL
- Embedded in host language programs to be precompiled
- As part of a procedure in an SQL module
- In dynamic SQL as a statement to be dynamically executed

Format



Usage Notes

Beginning with this release of Oracle Rdb, the simple CASE statement allows you to provide a list of literal values for every WHEN clause, instead of a single value as was supported in previous versions of Oracle Rdb.

Examples

Example 1 Using a List of Literal Values with the Case Statement

```
SQL> DECLARE :CODE CHAR(4);
SQL> BEGIN
cont>    JOB_LOOP:
cont>        FOR :JOBFOR
cont>          AS EACH ROW OF
cont>          SELECT * FROM JOBS JOB
cont>          DO
cont>          SET :CODE = :jobfor.JOB_CODE;
cont>            CASE :CODE
cont>              WHEN 'ASCK' THEN
cont>                 UPDATE JOBS
cont>                 SET MINIMUM_SALARY=10000
cont>                 WHERE JOB_CODE = :code;
cont>              WHEN 'ADMN', 'JNTR', 'SCTR' THEN
cont>                 UPDATE JOBS
cont>                 SET MINIMUM_SALARY=15000
cont>                 WHERE JOB_CODE = :code;
cont>              ELSE
cont>                 UPDATE JOBS
cont>                 SET MINIMUM_SALARY=:jobfor.MINIMUM_SALARY*1.1
cont>                 WHERE JOB_CODE=:code;
cont>            END CASE;
cont>        END FOR;
cont> END;
SQL>
```

COMMENT ON Statement

Adds or changes a comment about the following database objects:

- Catalog
- Collating sequence
- Column
- Constraint
- Database
- Domain
- Function
- Index
- Index partition
- Module
- Procedure
- Profile
- Role
- Schema
- Sequence
- Storage map
- Storage map partition
- Synonym
- Table
- Trigger
- User
- View

SQL displays the comments on these objects when you issue a SHOW statement.

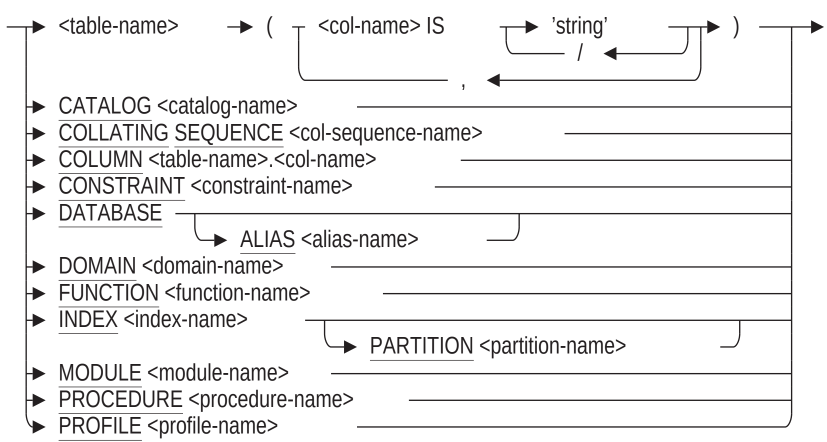
Environment

You can use the COMMENT ON statement:

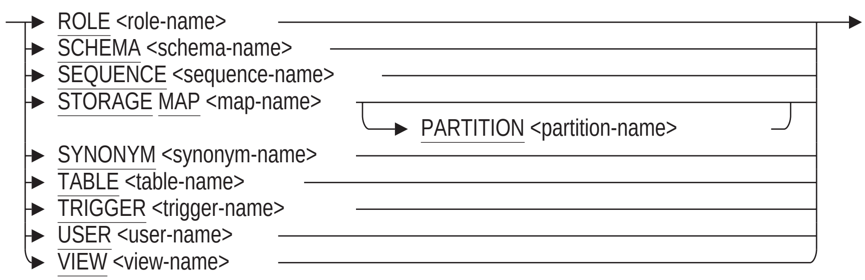
- In interactive SQL
- Embedded in host language programs to be precompiled
- As part of a procedure in an SQL module
- In dynamic SQL as a statement to be dynamically executed

Format

objects1 =



objects2 =



Arguments

COLLATING SEQUENCE col-sequence-name

Names the collating sequence for which you want to create a comment. If the collating sequence is not in the default schema, you must qualify the collating sequence name in the COMMENT ON statement with an authorization identifier.

DATABASE

This statement writes the new comment to the database. If the ALIAS clause is omitted, the default database is used. The alias-name must be an alias specified by an ATTACH or CONNECT statement during this session.

FUNCTION function-name

Names the function for which you want to create a comment. If the function is not in the default schema, you must qualify the function name in the COMMENT ON statement with an authorization identifier.

INDEX index-name
INDEX index-name PARTITION partition-name

Names the index and, optionally, a partition in the named index, for which you want to create a comment. If the index is not in the default schema, you must qualify the index name in the COMMENT ON statement with an authorization identifier.

MODULE module-name
Names the module for which you want to create a comment. If the module is not in the default schema, you must qualify the module name in the COMMENT ON statement with an authorization identifier.

PROCEDURE procedure-name
Names the procedure for which you want to create a comment. If the procedure is not in the default schema, you must qualify the procedure name in the COMMENT ON statement with an authorization identifier.

ROLE role-name
Names the role for which you want to create a comment. If the role is not in the default schema, you must qualify the role name in the COMMENT ON statement with an authorization identifier.

SEQUENCE sequence-name
Names the sequence for which you want to create a comment. If the sequence is not in the default schema, you must qualify the sequence name in the COMMENT ON statement with an authorization identifier.

STORAGE MAP map-name
STORAGE MAP map-name PARTITION partition-name
Names the storage map and, optionally, a vertical or horizontal partition within that storage map, for which you want to create a comment. If the storage map is not in the default schema, you must qualify the storage map name in the COMMENT ON statement with an authorization identifier.

SYNONYM synonym-name
This performs the same function as the ALTER SYNONYM ... COMMENT IS syntax. The synonym-name must be the name of an existing synonym. A database alias can be used to select a database other than the default database alias.

TRIGGER trigger-name
Names the trigger for which you want to create a comment. If the trigger is not in the default schema, you must qualify the trigger name in the COMMENT ON statement with an authorization identifier.

USER user-name
Names the user (created with the CREATE USER statement) for which you want to create a comment. If the user is not in the default schema, you must qualify the user name in the COMMENT ON statement with an authorization identifier.

VIEW view-name

Names the view for which you want to create a comment. If the view is not in the default schema, you must qualify the view name in the COMMENT ON statement with an authorization identifier.

table-name (col-name ’string’)

Names the table and the column or columns in that table for which you want to create a comment.

Usage Notes

No new usage notes.

Examples

Example 1 Adding a Comment to a Trigger

```
SQL> COMMENT ON TRIGGER EMPLOYEE_ID_CASCADE_DELETE IS
cont> 'When an employee is deleted from EMPLOYEES,’/
cont> 'delete corresponding records from the other tables in the database.';
SQL> SHOW TRIGGER EMPLOYEE_ID_CASCADE_DELETE
     EMPLOYEE_ID_CASCADE_DELETE
  Source:
 EMPLOYEE_ID_CASCADE_DELETE
                       BEFORE DELETE ON EMPLOYEES
                       (DELETE FROM DEGREES D WHERE D.EMPLOYEE_ID =
                        EMPLOYEES.EMPLOYEE_ID)
                          FOR EACH ROW
                       (DELETE FROM JOB_HISTORY JH WHERE JH.EMPLOYEE_ID =
                        EMPLOYEES.EMPLOYEE_ID)
                          FOR EACH ROW
                       (DELETE FROM SALARY_HISTORY SH WHERE SH.EMPLOYEE_ID =
                        EMPLOYEES.EMPLOYEE_ID)
                          FOR EACH ROW
                     -- Also, if an employee is terminated and that employee
                     -- is the manager of a department, set the manager_id
                     -- null for that department.
                       (UPDATE DEPARTMENTS D  SET D.MANAGER_ID = NULL
                        WHERE D.MANAGER_ID = EMPLOYEES.EMPLOYEE_ID)
                          FOR EACH ROW.
.
.
.
Comment:       When an employees is deleted from EMPLOYEES,
```

```
delete corresponding records from the other tables in the
database.
```

Example 2 Adding Comments to Multiple Columns in a Table

```
SQL> COMMENT ON JOBS (JOB_CODE is 'Required column',
cont> WAGE_CLASS is 'Valid values are: 1, 2, 3, or 4');
SQL> SHOW TABLE (COLUMNS) JOBS;
Information for table JOBS
Columns for table JOBS:
Column Name                     Data Type        Domain
-----------                     ---------        ------
JOB_CODE                        CHAR(4)          JOB_CODE
 Comment:       Required column
 Missing Value: None
WAGE_CLASS                      CHAR(1)          WAGE_CLASS
 Comment:       Valid values are: 1, 2, 3, or 4
JOB_TITLE                       CHAR(20)         JOB_TITLE
 Missing Value: None
MINIMUM_SALARY                  INTEGER(2)       SALARY
MAXIMUM_SALARY                  INTEGER(2)       SALARY
```

Example 3 Adding a Comment to a Database

The following example shows adding a comment to the database specified by the alias B:

```
SQL> COMMENT ON DATABASE
cont>     ALIAS B
cont>     IS  'any old comment will do'
cont>     /   'as long as it is descriptive.';
```

The SHOW DATABASE statement displays the comment as part of the list of attributes for the database. For example:

```
Alias B:
    Oracle Rdb database in file db$:mf_personnel_sql
        Multischema mode is disabled
        Number of users:               50
        Number of nodes:               16
.
.
.
        Dictionary Not Required
        ACL based protections
 Comment:       any old comment will do
```

```
                as long as it is descriptive.
Storage Areas in database with alias B
     DEPARTMENTS
```

Example 4 Changing the Comment of a Synonym

```
SQL> COMMENT ON SYNONYM mytable
cont>   IS 'a more descriptive name';
```

Compound Statement

Allows you to include more than one SQL statement in an SQL module procedure or in an embedded SQL program. Only by defining a compound statement can you put multiple SQL statements in a procedure. Procedures that contain one or more compound statements are called **multistatement procedures**.

In contrast, a simple statement can contain a single SQL statement only. Procedures that contain a single SQL statement are called **simple statement procedures**. See the *Oracle Rdb7 Guide to SQL Programming* for a description of simple statement procedures and how you use them in SQL application programming.

A compound statement and a simple statement differ not just in the number of SQL statements they can contain. A compound statement:

- Can include only a subset of the SQL statements allowed in a simple statement procedure. (See the compound-use-statement syntax diagram for a list of these valid statements.)
- Can include control flow statements, much like those you can use in a host language program. (See the control-statement syntax diagrams for a list of control flow statements allowed in a compound statement.)
- Can include transaction management statements, such as ROLLBACK and COMMIT.
- Can include local variables.
- Can control atomicity.
- Can reference only one alias, because each compound statement represents a single Oracle Rdb request.

See the *Oracle Rdb7 Guide to SQL Programming* for a conceptual description of compound statements and their relationship to multistatement procedures.

Environment

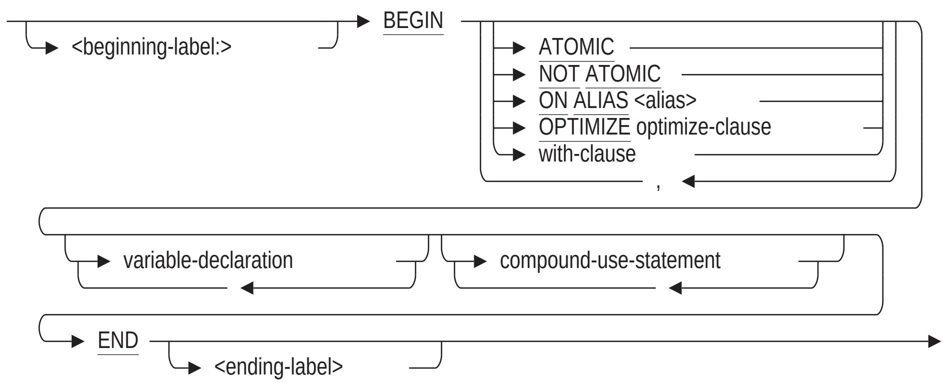
You can use a compound statement:

- In interactive SQL, as a way to test syntax and prototype compound statements for use with programs
- In embedded SQL, as part of a host language program to be processed with the SQL precompiler

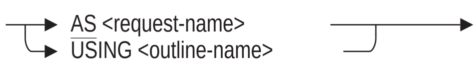
- In SQL module language, as part of a multistatement procedure in an SQL module file to be processed with the SQL module processor
- In dynamic SQL, to prepare and execute compound statements

Format

compound-statement =

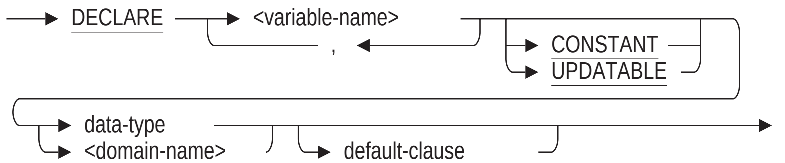


optimize-clause=

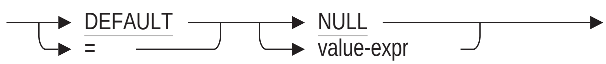


with-clause =

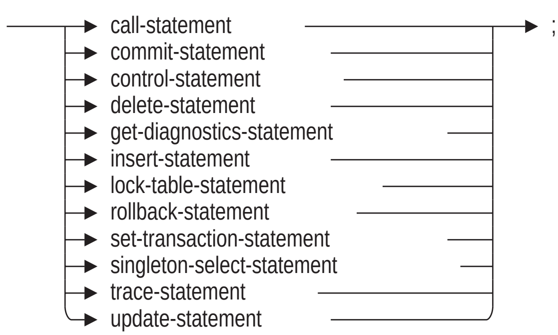
variable-declaration =



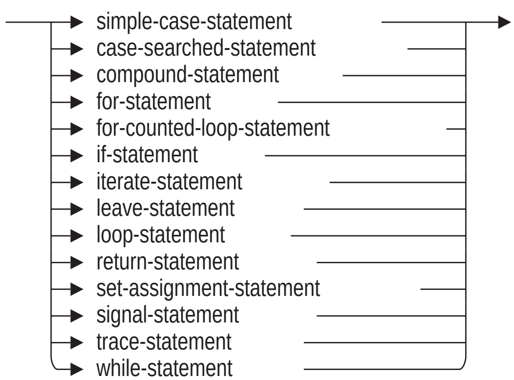
default-clause =



compound-use-statement =



control-statement =



Arguments

OPTIMIZE AS request-name
Assigns a name to the compound statement. You can specify this clause only on the outermost BEGIN keyword of the compound statement being assigned a name.

OPTIMIZE USING outline-name
Names the query outline to be used with the compound statement, even if the outline ID for the query and for the outline are different. You can specify this clause only on the outermost BEGIN keyword of the named compound statement.

lock-table-statement
See the LOCK TABLE Statement for a complete description.

simple-case-statement
See the CASE (Simple) Control Statement for a complete description.

case-searched-statement
See the CASE (Searched) Control Statement for a complete description.

for-counted-loop-statement
See the FOR (Counted) Control Statement for a complete description.

iterate-statement
See the ITERATE Control Statement for a complete description.

while-statement

See the WHILE Control Statement for a complete description.

Usage Notes

- If an outline exists, Oracle Rdb will use the outline specified in the OPTIMIZE USING clause unless one or more of the directives in the outline cannot be followed. SQL issues an error message if the existing outline cannot be used.
- If you specify the name of an outline that does not exist, Oracle Rdb compiles the query, ignores the outline name, and searches for an existing outline with the same outline ID as the query. If an outline with the same outline ID is found, Oracle Rdb attempts to execute the query using the directives in that outline. If an outline with the same outline ID is not found, the optimizer selects a strategy for the query for execution.
- See the *Oracle Rdb7 Guide to Database Performance and Tuning* for more information regarding query outlines.

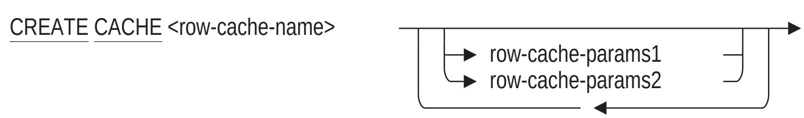
CREATE CACHE Clause

Creates a row cache area that allows frequently referenced rows to remain in memory even when the associated page has been transferred back to disk. This saves memory usage, because only the more recently referenced rows are cached, instead of the entire buffer.

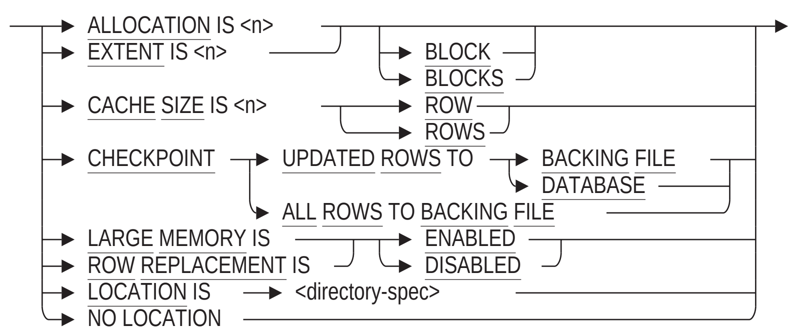
Environment

You can use the CREATE CACHE clause only within a CREATE DATABASE or IMPORT statement.

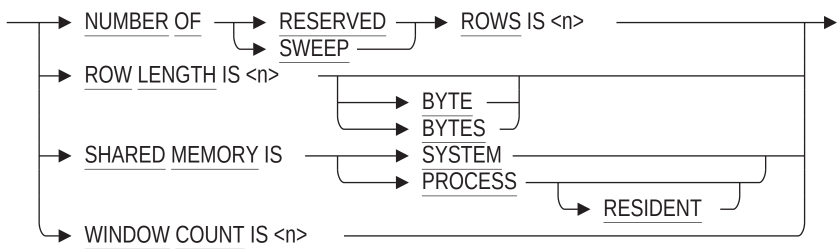
Format



row-cache-params1 =



row-cache-params2 =



Arguments

CACHE row-cache-name
Creates a row cache.

ALLOCATION IS n BLOCK
ALLOCATION IS n BLOCKS
Specifies the initial allocation of the row cache backing file (.rdc) to which cached rows are written during a checkpoint operation.

If the ALLOCATION clause is not specified, the default allocation in blocks is approximately 40 percent of the CACHE SIZE for this cache.

This clause is ignored if the row cache is defined to checkpoint to the database.

EXTENT IS n BLOCK
EXTENT IS n BLOCKS
Specifies the file extent size for the row cache backing file (.rdc).

If the EXTENT clause is not specified, the default number of blocks is the CACHE SIZE multiplied by 127.

This clause is ignored if the row cache is defined to checkpoint to the database.

CACHE SIZE IS n ROW
CACHE SIZE IS n ROWS
Specifies the number of rows allocated to the row cache. As the row cache fills, rows more recently referenced are retained in the row cache, while those not referenced

recently are discarded. Adjusting the allocation of the row cache helps to retain important rows in memory. If not specified, the default is 1000 rows.

The product of the CACHE SIZE and the ROW LENGTH settings determines the amount of memory required for the row cache. (Some additional overhead and rounding up to page boundaries are performed by the database system.) The row cache is shared by all processes attached to the database.

CHECKPOINT ALL ROWS TO BACKING FILE
CHECKPOINT UPDATED ROWS TO BACKING FILE
CHECKPOINT UPDATED ROWS TO DATABASE

Specifies the source records and target for checkpoint operations for the row cache. If ALL ROWS is specified, then the records written during each checkpoint operation are both the modified and the unmodified rows in the row cache. If UPDATED ROWS is specified, then just the modified rows in the row cache are checkpointed each time.

If the target of the checkpoint operation is BACKING FILE, then the row cache server (RCS) process writes the row cache entries to the backing (.rdc) files. The row cache LOCATION, ALLOCATION, and EXTENT clauses are used to create the backing files. Upon recovery from a node failure, the database recovery process is able to repopulate the row caches in memory from the rows found in the backing files.

If the target is DATABASE, then the updated rows (only UPDATED ROWS is allowed) are written back to the database. The row cache LOCATION, ALLOCATION, and EXTENT clauses are ignored. Upon recovery from a node failure, the database recovery process has no data on the contents of the row cache. Therefore, it does not repopulate the row caches in memory.

This CHECKPOINT clause overrides the database-level CHECKPOINT clause.

LARGE MEMORY IS ENABLED
LARGE MEMORY IS DISABLED

Specifies whether or not large memory is used to manage the row cache. Very large memory (VLM) allows Oracle Rdb to use as much physical memory as is available. It provides access to a large amount of physical memory through small virtual address windows.

Use LARGE MEMORY IS ENABLED only when both of the following are true:

- You have enabled row caching.
- You want to cache large amounts of data, but the cache does not fit in the virtual address space.

The default is DISABLED.

ROW REPLACEMENT IS ENABLED
ROW REPLACEMENT IS DISABLED

Specifies whether or not Oracle Rdb replaces rows in the cache. When you use the ROW REPLACEMENT IS ENABLED clause, rows are replaced when the row cache becomes full.

When you use the ROW REPLACEMENT IS DISABLED clause, rows are not replaced when the cache is full. The type of row replacement policy you use depends upon the application requirements for each cache.

The default is the ROW REPLACEMENT IS ENABLED clause.

LOCATION IS directory-spec

Specifies the name of the directory to which row cache backing file information is written. The database system generates a file name (row-cache-name.rdc) automatically for each row cache at checkpoint time. Specify a device name and directory name only and enclose this string in single quotation marks ('). By default, the location is the directory of the database root file. These .rdc files are permanent database files.

This LOCATION clause overrides a previously specified location at the database level.

This clause is ignored if the row cache is defined to checkpoint to the database.

NO LOCATION

Removes the location previously specified in a LOCATION IS clause for the row cache backing file. If you specify NO LOCATION, the row cache backing file location becomes the directory of the database root file.

This clause is ignored if the row cache is defined to checkpoint to the database.

NUMBER OF RESERVED ROWS IS n

Specifies the maximum number of cache rows that each user can reserve.

The default is 20 rows.

The number of reserved rows parameter is used also when searching for available slots in a row cache. The entire row cache is not searched on the initial pass. This parameter is used as the maximum number of rows that are searched for a free slot. If at least one free slot is found, the insert operation can proceed. If no free slots are found in this initial search, Oracle Rdb continues searching through the cache until it finds a free slot.

NUMBER OF SWEEP ROWS IS n

Specifies the number of modified cache rows that will be written back to the database to make space available in the row cache for subsequent transactions that insert rows into the cache. Oracle Corporation recommends that you initially specify the number of sweep rows to be between 10 and 30 percent of the total number of rows in the cache, then monitor performance and adjust the number of sweep rows if necessary. The default setting is 3000 rows.

ROW LENGTH IS n BYTE
ROW LENGTH IS n BYTES

Specifies the size of each row allocated to the row cache. Rows are not cached if they are longer than a row cache row. The ROW LENGTH is an aligned longword rounded up to the next multiple of 4 bytes.

If the ROW LENGTH clause is not specified, the default row length is 256 bytes. The maximum row length in a row cache area is 65535 bytes.

SHARED MEMORY IS SYSTEM
SHARED MEMORY IS PROCESS

Determines whether cache global sections are created in system space or process space. The default is SHARED MEMORY IS PROCESS.

When you use cache global sections created in the process space, you and other users share physical memory, and the OpenVMS Alpha operating system maps a row cache to a private address space for each user. As a result, all users are limited by the free virtual address range, and each uses a percentage of memory in overhead. If many users are accessing the database, the overhead can be high.

When many users are accessing the database, consider using SHARED MEMORY IS SYSTEM. This gives users more physical memory, because they share the system space of memory and there is none of the overhead associated with the process space of memory.

WINDOW COUNT IS n

Specifies the number of virtual address windows used by the LARGE MEMORY clause.

The window is a view into the physical memory used to create the very large memory (VLM) information. Because the VLM size may be larger than that which can be addressed by a 32-bit pointer, you need to view the VLM information through small virtual address windows.

You can specify a positive integer in the range from 10 through 65535. The default is 100 windows.

Usage Notes

- If the name of the row cache is the same as any logical area (for example a table name, index name, storage map name, RDB$SEGMENTED_STRINGS, RDB$SYSTEM_RECORD, and so forth), then this is a logical area cache and the named logical area is cached automatically. Otherwise, a storage area needs to be associated with the cache.
- The CREATE CACHE clause does not assign the row cache to a storage area. You must use the CACHE USING clause with the CREATE STORAGE AREA clause of the CREATE DATABASE statement or the CACHE USING clause with the ADD STORAGE AREA or ALTER STORAGE AREA clause of the ALTER DATABASE statement.
- The product of the CACHE SIZE and the ROW LENGTH settings determines the amount of memory required for the row cache (some additional overhead and rounding up to page boundaries are performed by the database system).
- The row cache is shared by all processes attached to the database on any one node.
- The following are requirements when using the row caching feature:
  - After-image journaling must be enabled.
  - Fast commit must be enabled.
  - Number of cluster nodes must equal 1.
- Use the SHOW CACHE statement to view information about a cache.

CREATE DATABASE Statement

Creates database system files, metadata definitions, and user data that comprise a database. The CREATE DATABASE statement lets you specify in a single SQL statement all data and privilege definitions for a new database. (You can also add definitions to the database later.) For information about ways to ensure good performance and data consistency, see the *Oracle Rdb7 Guide to Database Performance and Tuning.*

The many optional elements of the CREATE DATABASE statement make it very flexible. In its simplest form, the CREATE DATABASE statement creates database system files, specifies their names, and determines the physical characteristics of the database. Using the optional elements of the CREATE DATABASE statement, you can also specify:

- Whether the database created with CREATE DATABASE is **multifile** (separate database root file and storage area data file) or **single file** (combined database root file and storage area data file). Multifile databases can have many storage areas for user data, all separate from the database root file created by the CREATE DATABASE statement. Multifile databases include CREATE STORAGE AREA clauses in the CREATE DATABASE statement to create multiple storage area files for enhanced performance.

  The presence or absence of a CREATE STORAGE AREA clause in a CREATE DATABASE statement determines whether the database is single file or multifile. To create a multifile database, you must include a CREATE STORAGE AREA clause in the CREATE DATABASE statement. To create a single-file database, do not include a CREATE STORAGE AREA clause in the CREATE DATABASE statement.

  Statements that create multifile databases must include at least one CREATE STORAGE AREA clause.

- Values for various database root file parameters that override the system defaults. **Database root file** (.rdb) parameters describe characteristics of the database root file. Database root file parameters affect the entire database, whether it is a single-file or a multifile database.
- Values for storage area parameters that override system defaults. **Storage area** parameters describe characteristics of the database storage area files. In a single-file database, because the storage area data file is combined with the database root file, storage area parameters apply to a single storage area and

affect the entire database. In a multifile database, storage area parameters specify defaults for the main storage area, RDB$SYSTEM, and for any subsequent CREATE STORAGE AREA clauses within the CREATE DATABASE statement.

- Any number of database elements. Database elements are a CREATE CATALOG statement, a CREATE STORAGE AREA clause, or a GRANT statement. The CREATE DATABASE statements that create single-file databases cannot include a CREATE STORAGE AREA clause because this is specific to multifile databases. The CREATE DATABASE statements that create multifile databases must include at least one CREATE STORAGE AREA clause.

  Unlike the same statements outside a CREATE DATABASE statement, database elements do not use statement terminators. The first statement terminator that SQL encounters ends the CREATE DATABASE statement. Later CREATE or GRANT statements are not within the scope of the CREATE DATABASE statement.

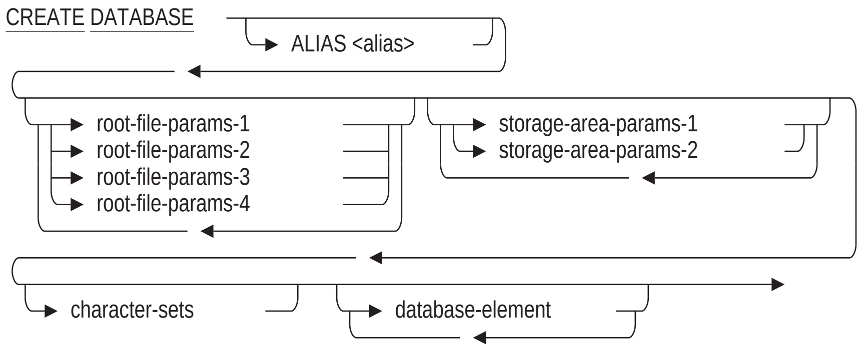
- The database default character set and national character set. For information regarding identifier character sets, database default character sets, and national character sets, see the *Oracle Rdb7 SQL Reference Manual*.

Environment

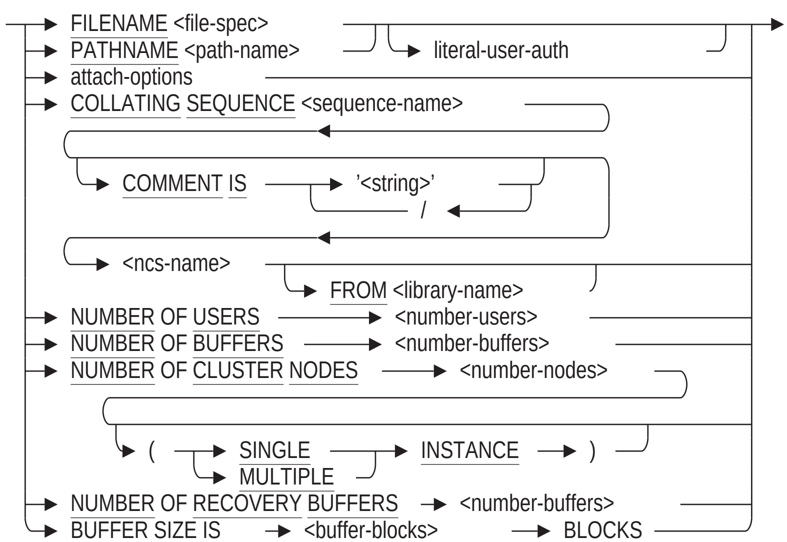
You can use the CREATE DATABASE statement:

- In interactive SQL
- Embedded in host language programs to be precompiled
- As part of a procedure in an SQL module
- In dynamic SQL as a statement to be dynamically executed

Format

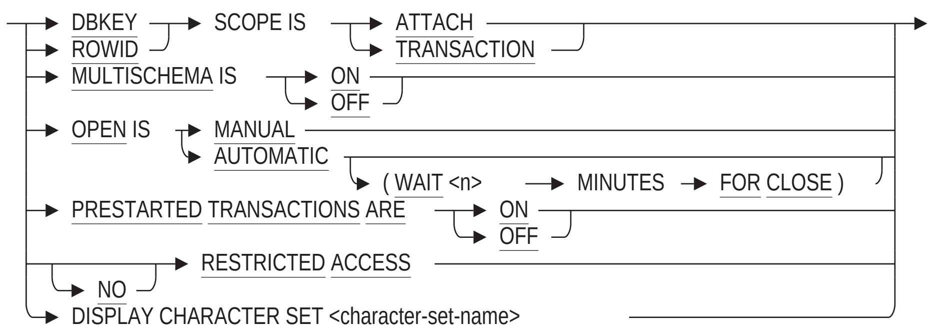


root-file-params-1 =

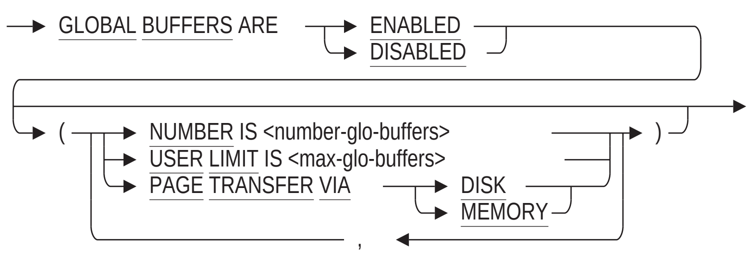


literal-user-auth =

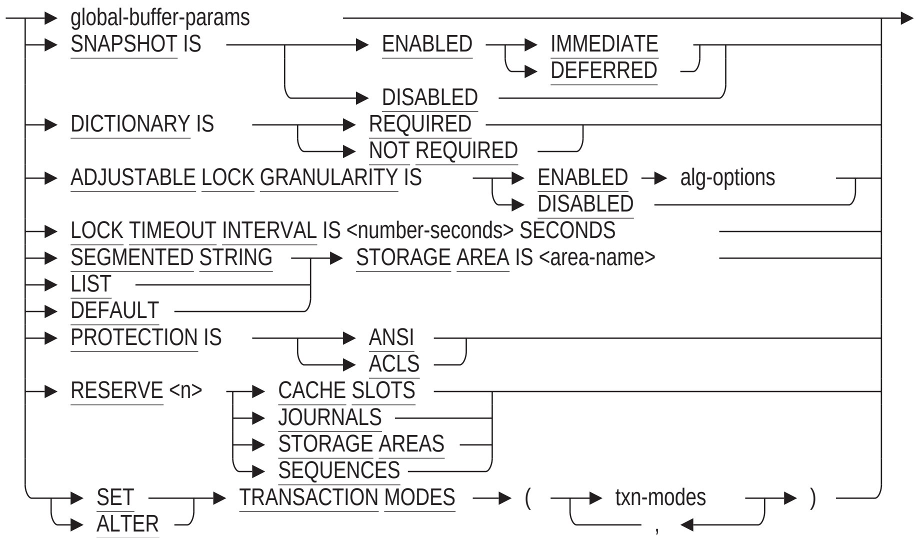
attach-options =



global-buffer-params=

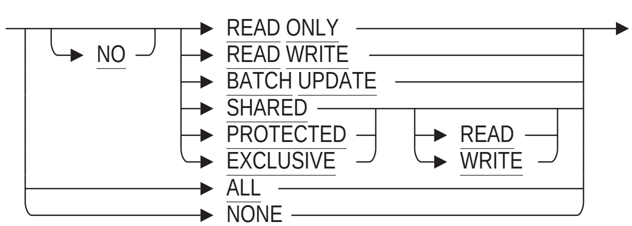


root-file-params-2 =

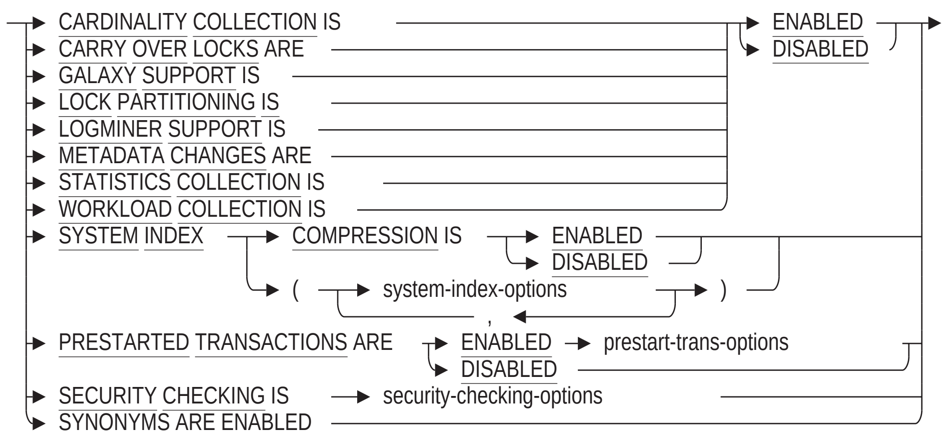


alg-options =

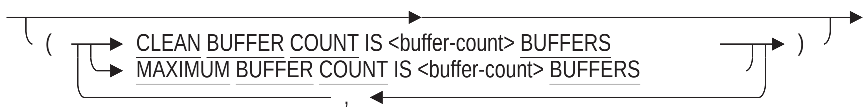
txn-modes =



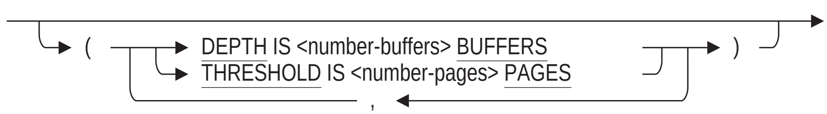
root-file-params-3 =



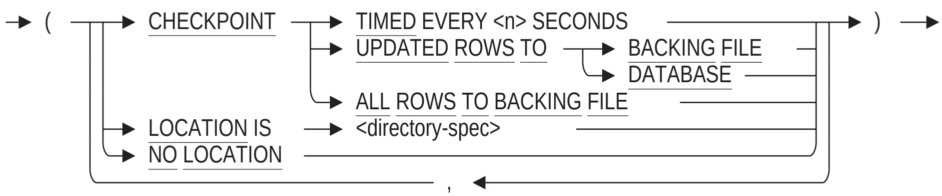
asynch-bat-wr-options =



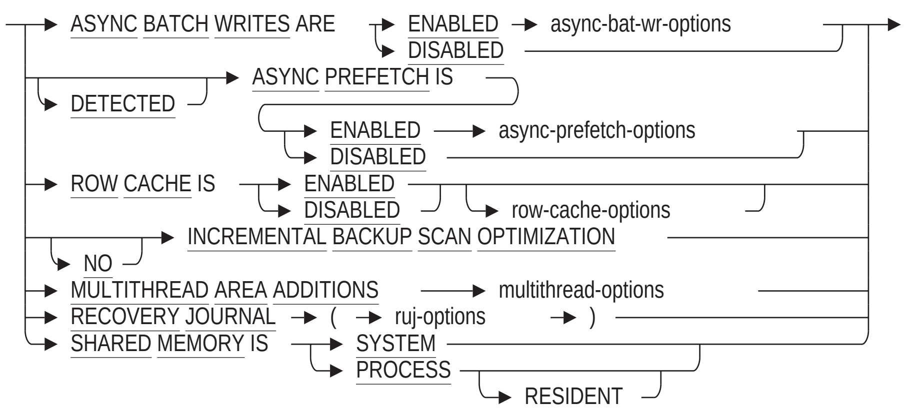
async-prefetch-options =



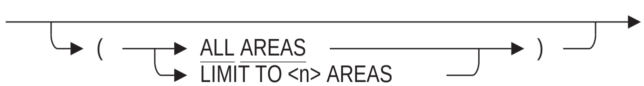
row-cache-options =



root-file-params-4 =



multithread-options =



ruj-options =

```
──┬─► LOCATION IS       ──► <directory-spec>      ──┬──►
  ├─► NO LOCATION       ────────────────────────────┤
  └─► BUFFER MEMORY IS  ─┬─► LOCAL  ────────────────┤
                         └─► GLOBAL ────────────────┘
```

storage-area-params-1 =

```
──┬─► ALLOCATION IS      ──► <number-pages>       ──► PAGES ──┬──►
  ├─► CACHE USING <row-cache-name>   ─────────────────────────┤
  ├─► NO ROW CACHE  ──────────────────────────────────────────┤
  ├─► extent-params ──────────────────────────────────────────┤
  ├─► INTERVAL IS        ──► <number-data-pages>  ────────────┤
  ├─► LOCKING IS         ─┬─► ROW ──┬──► LEVEL ───────────────┤
  │                       └─► PAGE ─┘                         │
  ├─► PAGE FORMAT IS     ─┬─► UNIFORM ─┬──────────────────────┤
  │                       └─► MIXED ───┘                      │
  └─► PAGE SIZE IS       ──► <page-blocks>        ──► BLOCKS ─┘
```

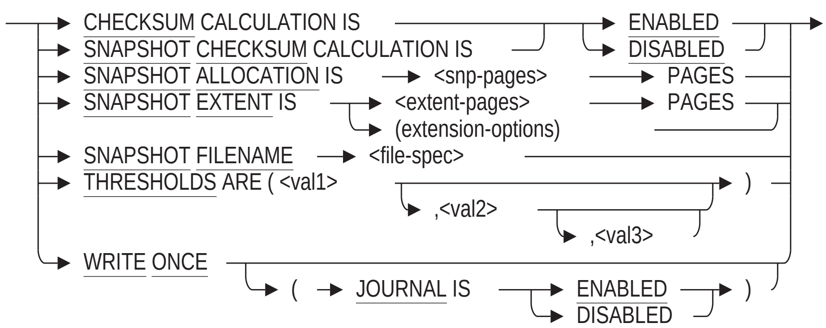
extent-params =

```
──┬─► EXTENT IS  ─┬─► ENABLED  ──────────────────────┬─┬──►
  │               ├─► DISABLED ──────────────────────┤ │
  │               ├─► <extent-pages>    ──► PAGES  ──┤ │
  │               └─► (extension-options) ───────────┘ │
  └───────────────────────◄────────────────────────────┘
```

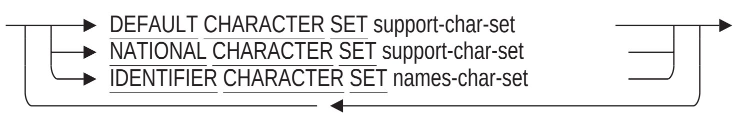
extension-options =

```
──► MINIMUM OF <min-pages> PAGES,    ──┐
┌──────────────────────────────────────┘
└─► MAXIMUM OF <max-pages> PAGES,    ──┐
┌──────────────────────────────────────┘
└─► PERCENT GROWTH IS <growth>     ──────►
```

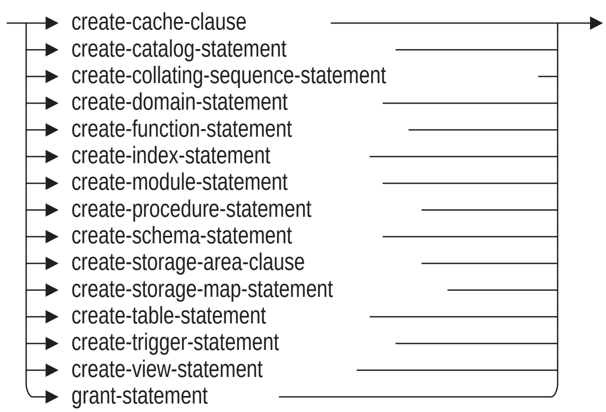
storage-area-params-2 =



character-sets =



database-element =



Arguments

RESERVE n CACHE SLOTS

Specifies the number of row caches for which slots are reserved in the database.

You can use the RESERVE CACHE SLOTS clause to reserve slots in the database root file for future use by the ADD CACHE clause.

Row caches can be added only if row cache slots are available. Slots become available after you issue a DROP CACHE clause or a RESERVE CACHE SLOTS clause.

You cannot decrease the number of reserved slots for row caches after you issue the RESERVE clause. If you reserve 10 slots and later reserve 5 slots, you have a total of 15 reserved slots for row caches.

Reserving row cache slots is an offline operation (requiring exclusive database access).

RESERVE n SEQUENCES

Specifies the number of sequences for which slots are reserved in the database. Sequences are reserved in multiples of 32. Thus, if you specify a value less than 32 for n, 32 slots are reserved. If you specify a value of 33, 64 slots are reserved, and so on.

You can use the RESERVE SEQUENCES clause to reserve slots in the database root file for future use by the CREATE SEQUENCE statement. Sequences can be created only if sequence slots are available. Slots become available after a DROP SEQUENCE statement or a RESERVE SEQUENCES clause of the ALTER DATABASE statement is executed.

The number of reserved slots for sequences cannot be decreased.

If you do not specify the RESERVE SEQUENCES clause, the default number of sequence slots is 32.

ROW CACHE IS ENABLED
ROW CACHE IS DISABLED

Specifies whether or not row caching is enabled. When a database is created or is converted from a previous version of Oracle Rdb without specifying row cache support, the default is ROW CACHE IS DISABLED. Enabling row cache support does not affect database operations until you create and assign a row cache area to one or more storage areas.

When the row caching is disabled, all previously created and assigned caches remain and will be available if row caching is enabled again.

CHECKPOINT TIMED EVERY n SECONDS

For the row-cache-options clause, specifies the frequency with which the row cache server (RCS) process checkpoints the contents of the row caches back to disk. Note that the RCS process does not use the checkpoint frequency options of the FAST COMMIT clause.

The frequency of RCS checkpointing is important in determining how much of an .aij file must be read during a recovery operation following a node failure. It also affects the frequency with which marked records get flushed back to the database, for those row caches that checkpoint to the database. The default is every 15 minutes (900 seconds).

CHECKPOINT UPDATED ROWS TO BACKING FILE
CHECKPOINT UPDATED ROWS TO DATABASE
CHECKPOINT ALL ROWS TO BACKING FILE

Specifies the default source and target during checkpoint operations for all row caches. If ALL ROWS is specified, then the source records written during each checkpoint operation are both the modified and the unmodified rows in a row cache. If UPDATED ROWS is specified, then just the modified rows in a row cache are checkpointed each time.

If the target of the checkpoint operation is BACKING FILE, then the RCS process writes the source row cache entries to the backing (.rdc) files. The row cache LOCATION, ALLOCATION, and EXTENT clauses are used to create the backing files. Upon recovery from a node failure, the database recovery process is able to repopulate the row caches in memory from the rows found in the backing files.

If the target is DATABASE, then the target rows are written back to the database. The row cache LOCATION, ALLOCATION, and EXTENT clauses are ignored. Upon recovery from a node failure, the database recovery process has no data on the contents of the row cache. Therefore, it does not repopulate the row caches in memory.

The CHECKPOINT clause of the CREATE CACHE, ADD CACHE, or ALTER CACHE clause overrides this database-level CHECKPOINT clause.

LOCATION IS directory-spec

Specifies the name of the default directory to which all row cache backing files (.rdc) are written. The database system generates a file name automatically for each row cache backing file it creates when the RCS process first starts up. Specify a device name and directory name only and enclose the string within single quotation marks ('). By default, the location is the directory of the database root file.

The LOCATION clause of the CREATE CACHE, ADD CACHE, or ALTER CACHE clause overrides this location, which is the default for the database.

NO LOCATION

Removes the default database location previously specified in a LOCATION IS clause for the row cache backing file (.rdc). If you specify NO LOCATION, then the row cache backing file location becomes the directory of the database root file.

The LOCATION clause of the CREATE CACHE, ADD CACHE, or ALTER CACHE clause overrides this location, which is the default for the database.

RECOVERY JOURNAL (LOCATION IS directory-spec)

Specifies the location, including device and directory, in which the recovery-unit journal (.ruj) file is written. Do not include network node names, file names (including file type or version on OpenVMS) or process-concealed logical names.

If this clause is omitted, the default directory location is the current device that the database root file uses and the special directory [RDM$RUJ]. You can use the RDMS$RUJ logical name to override this clause.

RECOVERY JOURNAL (NO LOCATION)
Removes a location previously defined by a RECOVERY JOURNAL LOCATION IS clause. This causes the recovery journal to revert to the default location.

RECOVERY JOURNAL (BUFFER MEMORY IS LOCAL)
RECOVERY JOURNAL (BUFFER MEMORY IS GLOBAL)
Specifies whether RUJ buffers will be allocated in global or local memory.

The RUJ buffers used by each process are normally allocated in local virtual memory. With the introduction of row caching, these buffers now can be assigned to a shared global section (global memory) on OpenVMS, so that the recovery process can process this in-memory buffer and possibly avoid a disk access.

You can define this buffer memory to be global to improve row caching performance for recovery. If row caching is disabled, then buffer memory is always local.

SECURITY CHECKING
Traditionally, Oracle Rdb has performed security checking using the operating system security layer (for example, the UIC and rights identifiers of the OpenVMS operating system).

The access control list (ACL) information stored in the database contains a granted privilege mask and a set of users represented by a unique integer (for example, a UIC).

There are three modes of security checking:

- SECURITY CHECKING IS EXTERNAL

  This is the default. External security checking recognizes users (created with the SQL CREATE USER statement) as operating system user identification codes (UICs) and roles as special rights identifiers or groups.

- SECURITY CHECKING IS INTERNAL (ACCOUNT CHECK IS ENABLED)

  The ACCOUNT CHECK clause ensures that Oracle Rdb validates the current user with the user name (as defined with an SQL CREATE USER statement) stored in the database. This prevents different users with the same name from accessing the database. Therefore, this clause might prevent a breach in security.

Note: The ACCOUNT CHECK IS ENABLED clause on OpenVMS does not check the SID, but does force the user session to have the same user name and UIC as recorded in the database.

If you specify the ACCOUNT CHECK IS DISABLED clause, then a user with a matching UIC (also called a profile-id) is considered the same as the user even if his or her user name is different. This allows support for multiple OpenVMS users with the same UIC.

- SECURITY CHECKING IS INTERNAL (ACCOUNT CHECK IS DISABLED)

  This syntax means that the assigned SID is ignored during database attach.

CACHE USING row-cache-name

Assigns the named row cache as the default physical row cache for all storage areas in the database. All rows stored in each storage area are cached, regardless of whether they consist of table data, segmented string data, or special rows such as index nodes.

You must create the row cache before terminating the CREATE DATABASE statement. For example:

```
SQL> CREATE DATABASE FILENAME test_db
cont> ROW CACHE IS ENABLED
cont> CACHE USING test1
cont> CREATE CACHE test1
cont>    CACHE SIZE IS 100 ROWS
cont> CREATE STORAGE AREA area1;
```

If you do not specify the CACHE USING clause or the NO ROW CACHE clause, the NO ROW CACHE clause is the default for the database.

You can override the database default row cache by specifying either the CACHE USING clause after the CREATE STORAGE AREA clause, or by altering the database and storage area later to assign a new row cache. Only one physical area row cache is allowed for each storage area.

Note that you can have multiple row caches containing rows for a single storage area by defining logical area row caches, where the row cache name matches the name of a table or index.

NO ROW CACHE

Specifies that the database default is to not assign a row cache to all storage areas in the database. You cannot specify the NO ROW CACHE clause if you specify the CACHE USING clause.

Alter the storage area and name a row cache to override the database default. Only one row cache is allowed for each storage area.

If you do not specify the CACHE USING clause or the NO ROW CACHE clause, NO ROW CACHE is the default for the database.

create-cache-clause

See the CREATE CACHE Clause for a description of this clause and its arguments.

Usage Notes

- The number of reserved slots for sequences cannot be decreased.
- If you do not specify the RESERVE SEQUENCE SLOTS clause, the default number of sequence slots is 32.
- The RDB$PROFILES system relation is used to record users and roles created with the CREATE USER and CREATE ROLE statements. When a database is created, the creator is automatically added as a user.
- The GRANT statement may reference operating system users or groups prior to those users or roles being created in the database. In this case, Oracle Rdb automatically creates users and roles that correspond to the IDENTIFIED EXTERNALLY clause of the CREATE USER or CREATE ROLE statements.

Examples

Example 1 Reserving Slots for Sequences

```
SQL> CREATE DATABASE FILENAME many_sequences
cont> RESERVE 320 SEQUENCES;
```

Example 2 Creating a Database with a Row Cache

```
SQL> CREATE DATABASE FILENAME sample
cont>   SNAPSHOT IS DISABLED
cont>   RESERVE 10 JOURNALS
cont>   RESERVE 10 STORAGE AREAS
cont>   RESERVE 5 CACHE SLOTS
cont>   SYSTEM INDEX COMPRESSION IS ENABLED
```

```
cont>   ROW CACHE IS ENABLED
cont>   WORKLOAD COLLECTION IS ENABLED
cont>   RESTRICTED ACCESS
cont>   STATISTICS COLLECTION IS DISABLED
cont>   CARDINALITY COLLECTION IS DISABLED
cont>   LOCKING IS ROW LEVEL
cont>   DEFAULT STORAGE AREA IS area1
cont>   OPEN IS AUTOMATIC (WAIT 5 MINUTES FOR CLOSE)
cont>   RECOVERY JOURNAL (LOCATION IS 'USER1:[JOURN]')
cont> CREATE CACHE area1
cont>   SHARED MEMORY IS PROCESS
cont>   ROW LENGTH IS 1000 BYTES
cont>   CACHE SIZE IS 204 ROWS
cont>   CHECKPOINT ALL ROWS TO BACKING FILE
cont>   NUMBER OF SWEEP ROWS IS 25
cont>   LOCATION IS 'USER1:[RCS]'
cont>   ALLOCATION IS 1 BLOCK
cont> CREATE STORAGE AREA area1;
SQL> SHOW DATABASE sample
SQL> DISCONNECT DEFAULT;
SQL> ATTACH DATABASE 'FILENAME sample';
SQL> show database *
Default alias:
    Oracle Rdb database in file sample
        Multischema mode is disabled
        Number of users:               50
        Number of nodes:               16
        Buffer Size (blocks/buffer):   6
        Number of Buffers:             20
        Number of Recovery Buffers:    20
        Snapshots are Disabled
        Carry over locks are enabled
        Lock timeout interval is 0 seconds
        Adjustable lock granularity is enabled (count is 3)
        Global buffers are disabled (number is 250, user limit is 5,
                  page transfer via disk)
        Journal fast commit is disabled
                ( checkpoint interval is 0 blocks,
                  checkpoint timed every 0 seconds,
                  no commit to journal optimization,
                  transaction interval is 256 )
        AIJ File Allocation:           512
        AIJ File Extent:               512
        Statistics Collection is DISABLED
        Unused Storage Areas:          10
```

```
Unused Journals:               10
        Unused Cache Slots:            5
        Unused Sequences:              32
        System Index Compression is ENABLED
        No Restricted Access
        Journal is Disabled
        Backup Server:   Manual
        Log Server:      Manual
        Overwrite:       Disabled
        Notification:    Disabled
        Asynchronous Prefetch is Enabled (depth is 5)
        Asynchronous Batch Write is Enabled (clean buffers 5, max buffers 4)
        Lock Partitioning is DISABLED
        Incremental Backup Scan Optim uses SPAM pages
        Shutdown Time is 60 minutes
        Workload Collection is Enabled
        Cardinality Collection is Disabled
        Metadata Changes are Enabled
        Row Cache is Enabled
        Row cache: No Location
        Row cache: checkpoint updated rows to backing file
        Detected Asynch Prefetch is Enabled (depth is 4, threshold is 4)
        Default Storage Area AREA1
        Mode is Open Automatic (Wait 5 minutes for close)
RUJ File Location RDBVMS_USER1:[STEWART]
        Database Transaction Mode(s) Enabled:
            ALL
        Shared Memory:         Process
        Dictionary Not Required
        ACL based protections
Storage Areas in database with filename sample
     AREA1                            Default storage area
     RDB$SYSTEM                       List storage area.
Journals in database with filename sample
     No Journals Found
Cache Objects in database with filename sample
     AREA1
```

CREATE DOMAIN Statement

Creates a domain definition.

A domain defines the set of values, character set, collating sequence, and formatting clause that a column in a table can have. The CREATE DOMAIN statement specifies the set of values by associating a data type with a domain name. The CREATE and ALTER TABLE statements can use the domain in column definitions.

There are two ways to specify a domain definition:

- With a domain name, data type, and any combination of the following optional clauses:
  - Default value
  - Stored name
  - Collating sequence
  - SQL and DATATRIEVE formatting clauses
- With the FROM clause and a repository path name that refers to a field already defined in the repository

When the CREATE DOMAIN statement executes, SQL adds the domain definition to the database.

If you attached to the database with the PATHNAME specification, the domain definition is also added to the repository.

You can refer to a domain instead of an SQL data type in the CREATE and ALTER TABLE statements and in formal parameter declarations in SQL module procedures. You can specify the same domain in many table definitions and in SQL module parameter declarations. If the domain has to change, you need only change that one domain definition (using the ALTER DOMAIN statement) to change all of the tables and SQL modules that refer to it. This ability makes it easier to keep applications consistent.

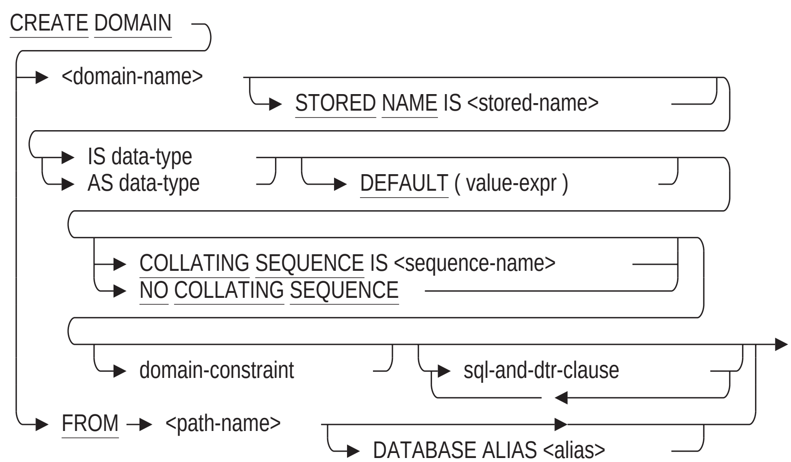
SQL lets you specify a character data type or national character data type when defining a domain. It also lets you specify whether the length of the domain is measured in characters or octets.

Environment

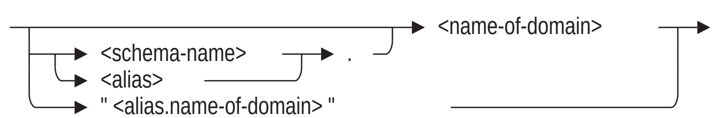
You can use the CREATE DOMAIN statement:

- In interactive SQL
- Embedded in host language programs to be precompiled
- As part of a procedure in an SQL module
- In dynamic SQL as a statement to be dynamically executed

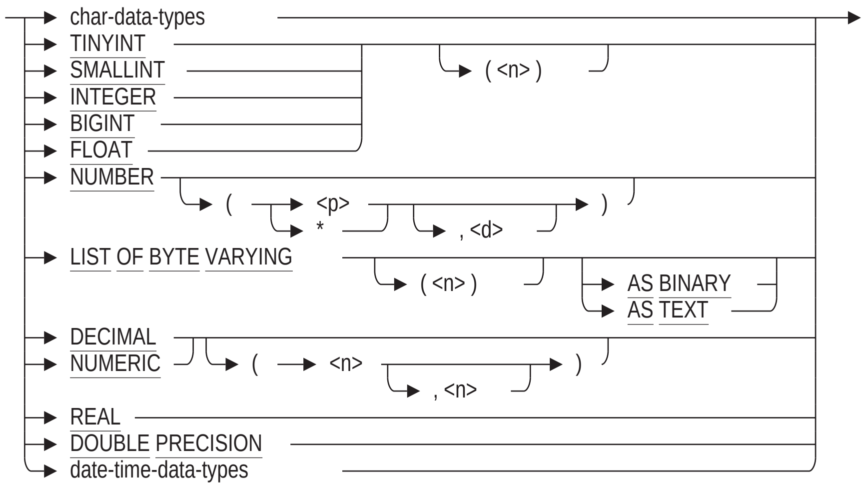
Format



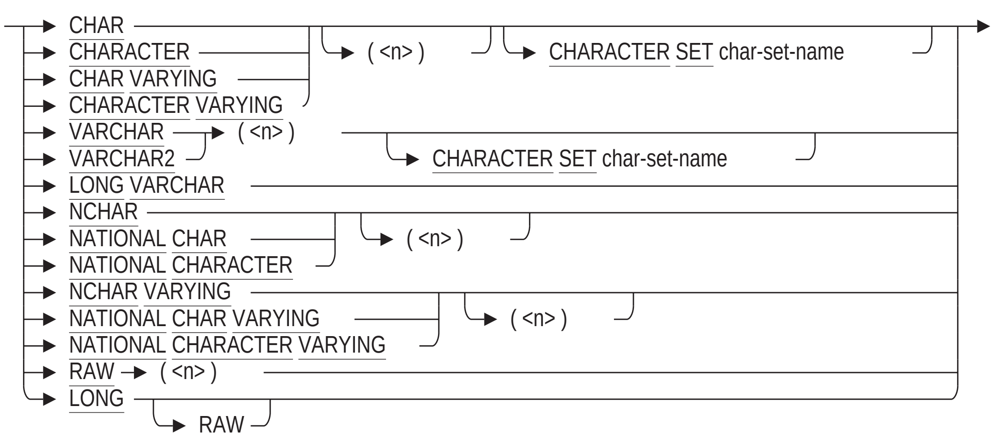
domain-name =



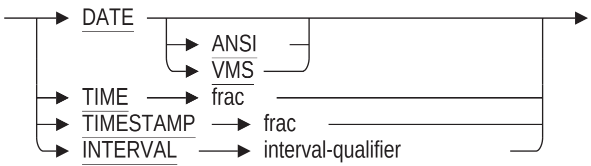
data-type =



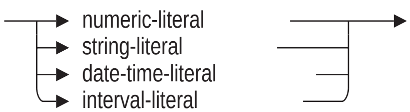
char-data-types =



date-time-data-types =

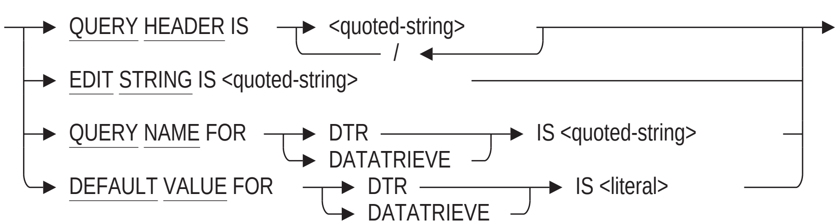


literal =



domain-constraint =

sql-and-dtr-clause =



Arguments

DEFAULT value-expr

A value to be stored in a column if the row that is inserted does not include a value for that column.

If you do not specify a default value, SQL assigns NULL as the default value.

Usage Notes

No new usage notes.

Examples

Example 1 Using the New NUMBER Data Type (Note the Implicit Conversion), and Including a COMMENT IS Clause

```
SQL> CREATE DOMAIN
cont>     MONEY
cont>     AS NUMBER (3)
cont>     DEFAULT 0.000
cont>     COMMENT IS 'use this to represent monetary amounts';
%SQL-F-FIELD_EXISTS, Domain MONEY already exists in this database or schema
SQL>
```

Example 2 Using the New Extended DEFAULT Syntax and Capitalizing the CURRENT_USER String When Used as a Default

```
SQL> CREATE DOMAIN
cont>     USER_SPEC
cont>     AS VARCHAR (40)
cont>     DEFAULT UPPER (SUBSTRING (current_user from 1 for 1))
cont>             || LOWER (SUBSTRING (current_user from 2));
```

CREATE INDEX Statement

Creates an index for a table. An index allows direct access to the rows in the table to avoid sequential searching.

You define an index by listing the columns in a table that make up the index. You can define more than one index for a table. The index can be made up of one column, or two or more columns. An index made up of two or more columns is called a **multisegmented index**.

Optional arguments to the CREATE INDEX statement let you specify:

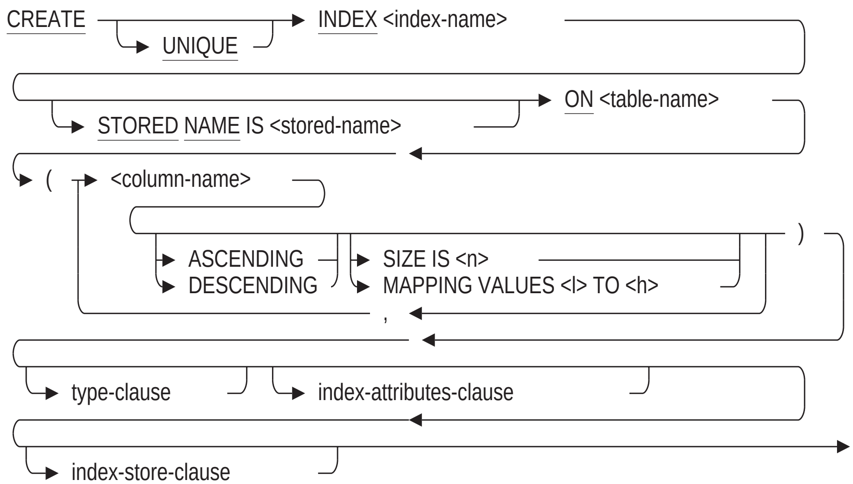
- The type of index structure (hashed, sorted nonranked, or sorted ranked)
- The names of a storage area or storage areas that contain the index
- Physical characteristics of a sorted index structure, such as index node size and the initial fullness percentage of each node
- Compression characteristics, including compressed key suffixes for text indexes and integer column compression for word or longword numeric columns
- Compression of space characters from text data types and of binary zeros from nontext data types
- Duplicates handling for sorted, ranked indexes
- Thresholds for the logical storage areas that contain the index
- Whether logging to the .ruj and .aij files is enabled or disabled for the create index operation
- A comment for the index definition

Environment

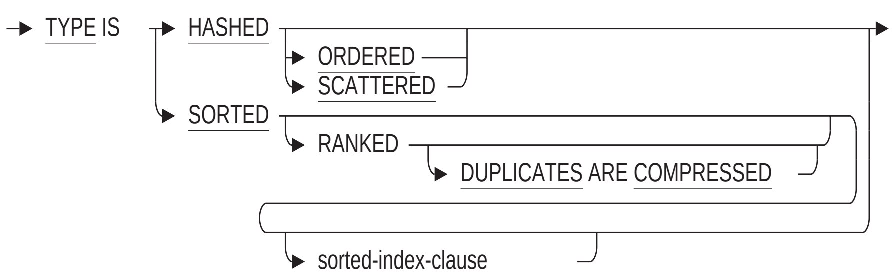
You can use the CREATE INDEX statement:

- In interactive SQL
- Embedded in host language programs to be precompiled
- As part of a procedure in an SQL module
- In dynamic SQL as a statement to be dynamically executed

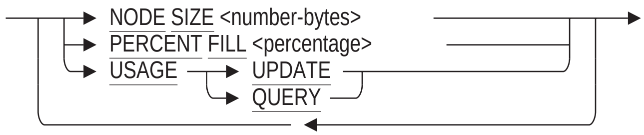
Format



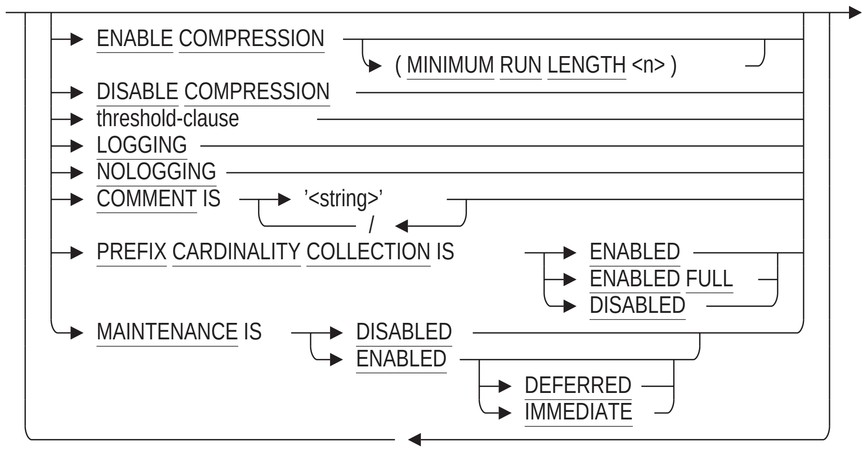
type-clause =



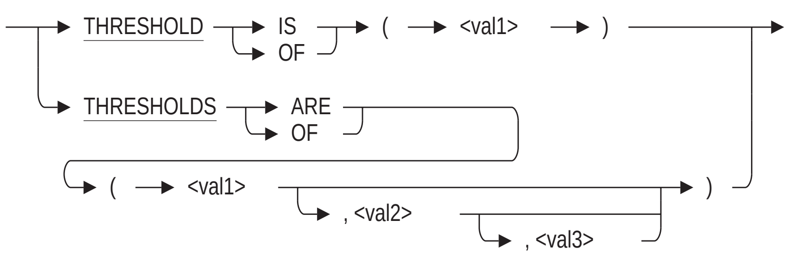
sorted-index-clause =



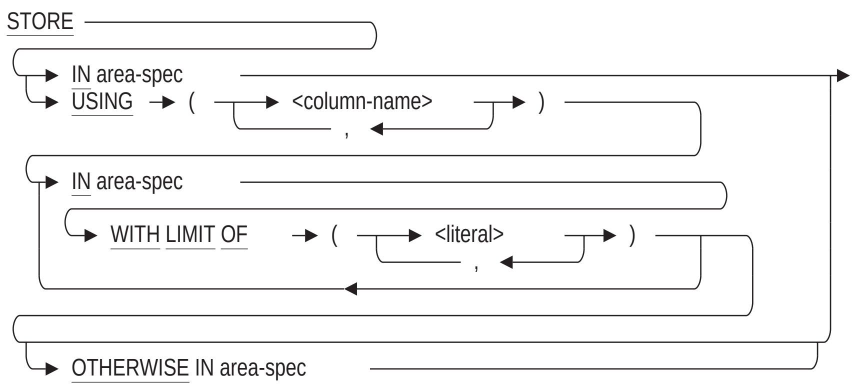
index-attributes-clause =



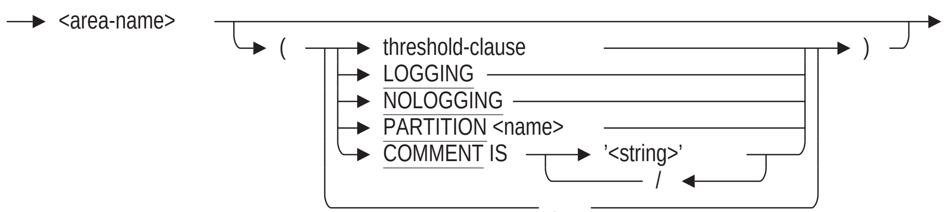
threshold-clause =



index-store-clause =



area-spec =



Arguments

threshold-clause

Specifies one, two, or three default threshold values for logical areas that contain the index in storage areas with uniform page formats. By setting threshold values, you can make sure that Oracle Rdb does not overlook a page with sufficient space to store compressed data. The threshold values (val1, val2, and val3) represent a fullness percentage on a data page and establish three possible ranges of guaranteed free space on the data pages.

If you use data compression, you should use logical area thresholds to obtain optimum storage performance.

You cannot specify the thresholds for the storage map attribute for any area that is a mixed page format. If you have a mixed page format, set the thresholds for the storage area using the ADD STORAGE AREA or CREATE STORAGE AREA clause of the ALTER DATABASE, CREATE DATABASE, or IMPORT statement.

For more information about SPAM pages, see the *Oracle Rdb7 Guide to Database Design and Definition*.

LOGGING
NOLOGGING
The LOGGING clause specifies that the partition should be logged in the recovery-unit journal file (.ruj) and after-image journal file (.aij).

The NOLOGGING clause specifies that the partition should not be logged in the recovery-unit journal file (.ruj) and after-image journal file (.aij).

If no store clause is used, then these attributes provide the setting for the CREATE INDEX statement.

The LOGGING and NOLOGGING clauses are mutually exclusive; specify only one. The LOGGING clause is the default.

See Section 2.2.8 for information on the advantages and disadvantages of specifying these clauses and for the implications that using the NOLOGGING keyword has on database recovery.

PARTITION name
Names the partition. The name can be a delimited identifier. Partition names must be unique within the index. If you do not specify this clause, Oracle Rdb generates a default name for the partition.

COMMENT IS ’string’
Adds a comment about the storage map definition for the index. SQL displays the text of the comment when it executes a SHOW INDEXES statement. Enclose the comment in single quotation marks (’) and separate multiple lines in a comment with a slash mark (/).

Usage Notes

- The CREATE INDEX statement supplies a default index node size if none is provided for a UNIQUE SORTED index or a SORTED RANKED index. Use the SQL SHOW INDEX or SHOW TABLE statement or the RMU Extract command to display the value of this default node size.

Examples

Example 1 Using the Index Attributes Clause

```
SQL> CREATE UNIQUE INDEX JOB_JOB_CODE
cont> ON JOBS (
cont> JOB_CODE
cont> ASC)
cont> TYPE IS SORTED
cont> THRESHOLDS ARE (75,83,90)
cont> ENABLE COMPRESSION
cont> NOLOGGING
cont> COMMENT IS 'Used for translation of job codes';
%RDB-W-META_WARN, metadata successfully updated with the reported warning
-RDMS-W-DATACMIT, unjournaled changes made; database may not be recoverable
SQL> -- SQL returned this message because the NOLOGGING attribute
SQL> -- was set.
```

Example 2 Creating an Index and Displaying the Default Node Size

```
SQL> -- Create a simple table upon which to define
SQL> -- some indexes
SQL>
SQL> CREATE TABLE TEST_INDEX_TABLE
cont>    (A CHAR(70),
cont>     B INTEGER);
SQL>
SQL> -- Default value is 430 bytes
SQL>
SQL> CREATE UNIQUE INDEX TEST_INDEX_DEF
cont>    ON TEST_INDEX_TABLE (A, B)
cont>    TYPE IS SORTED
cont>    USAGE UPDATE;
SQL>
SQL> SHOW TABLE (INDEX) TEST_INDEX_TABLE
Information for table TEST_INDEX_TABLE
TEST_INDEX_DEF                  with column A
                                and column B
  No Duplicates allowed
  Type is Sorted
  Compression is DISABLED
  Node size  430
  Percent fill  70
```

Example 3 Naming Partitions

```
SQL> -- Alter mf_personnel database to add three slots
SQL> -- for storage areas and then add three storage areas.
SQL> ALTER DATABASE FILENAME MF_PERSONNEL
cont> RESERVE 3 STORAGE AREAS;
%RDMS-W-DOFULLBCK, full database backup should be done to ensure future recovery
SQL> ALTER DATABASE FILENAME MF_PERSONNEL
cont> ADD STORAGE AREA WAGE_LOW;
SQL> ALTER DATABASE FILENAME MF_PERSONNEL
cont> ADD STORAGE AREA WAGE_MID;
SQL> ALTER DATABASE FILENAME MF_PERSONNEL
cont> ADD STORAGE AREA WAGE_HIGH;
SQL> ATTACH 'FILENAME MF_PERSONNEL.RDB';
SQL> -- Create an index on the JOBS table and name the partitions
SQL> CREATE INDEX WAGE_CLASS_IDX ON JOBS (WAGE_CLASS)
cont> TYPE IS SORTED
cont> STORE USING (WAGE_CLASS)
cont> IN WAGE_LOW (PARTITION WAGE_LOW) WITH LIMIT OF ('1')
cont> IN WAGE_MID (PARTITION WAGE_MID) WITH LIMIT OF ('3')
cont> OTHERWISE IN WAGE_HIGH (PARTITION WAGE_HIGH);
```

CREATE MODULE Statement

Defines a module as an object in an Oracle Rdb database. Stored with the module are its functions and procedures. A function or procedure that resides with the data in a database is called a **stored function** or **stored procedure**. Likewise, a module stored in a database is called a **stored module**. A **stored routine** refers to either a stored procedure or stored function.

You invoke a stored procedure with the CALL statement from a simple statement procedure in embedded SQL, SQL module language, or interactive SQL, or with the CALL statement from within a compound statement.

You invoke a stored function by specifying the function name in a value expression.

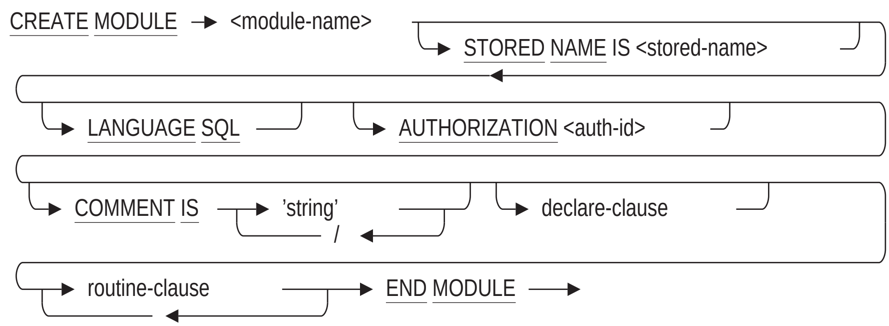
SQL uses the concept of a module as its mechanism for storing, showing, deleting, and granting and revoking privileges on stored routines within a database. This means that you cannot store, delete, or grant and revoke privileges on individual stored routines. Should you need to remove a stored routine, use the DROP FUNCTION routine-name CASCADE or DROP PROCEDURE routine-name CASCADE syntax of the Drop Routine statement.

Environment

You can use the CREATE MODULE statement in a simple statement procedure:

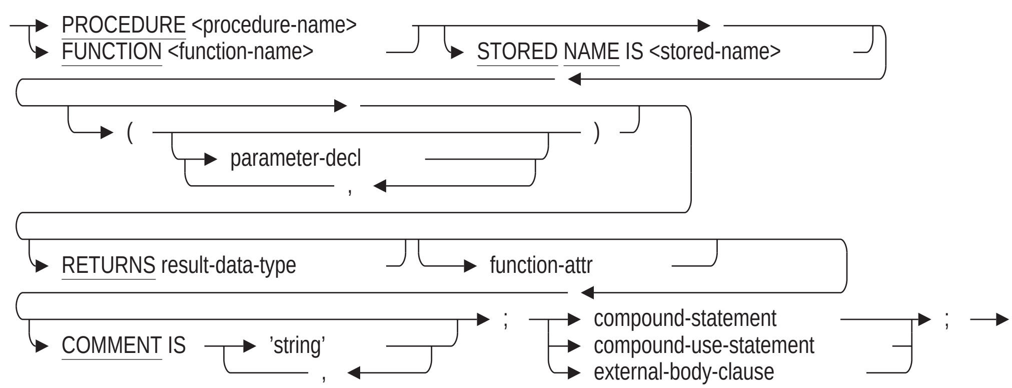
- In interactive SQL
- Embedded in host language programs to be precompiled
- As part of a procedure in an SQL module
- In dynamic SQL as a statement to be dynamically executed

Format

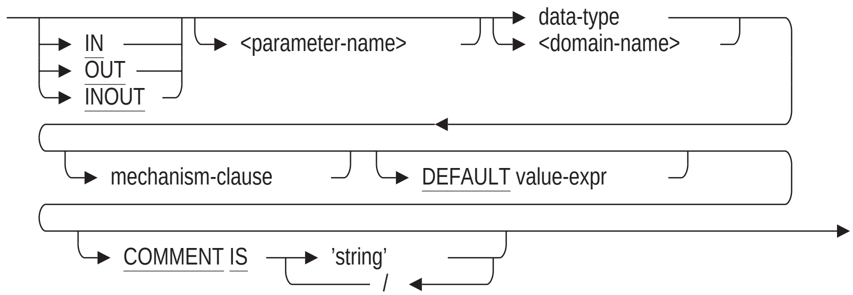


declare-clause =

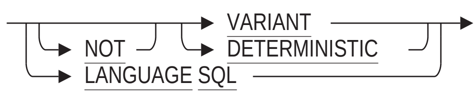
routine-clause =



parameter-decl =



function-attr =



Arguments

LANGUAGE SQL

The LANGUAGE keyword and the SQL argument signify that the procedures in a module are to be invoked by SQL statements, not a host language program.

With unstored procedures, the LANGUAGE keyword specifies the name of a host language; this identifies the host language in which the program calling a module's procedures is written.

Beginning with Oracle Rdb release 7.1, this clause is optional.

declare-variable-statement

Declares a global variable for the module. See the entry for the DECLARE Variable statement in the *Oracle Rdb7 SQL Reference Manual* for more information.

DEFAULT value-expr

Specifies the default value of a parameter for an external function or procedure defined with mode IN. If you omit this parameter or if the Call statement argument list or function invocation specifies the DEFAULT keyword, then the value-expr specified with this clause is used. The parameter uses NULL as the default if you do not specify a value expression explicitly. However, this results in a run-time error unless you defined the external routine with the RETURNS NULL ON NULL INPUT clause.

COMMENT IS ’string’

Adds a comment about the parameter. SQL displays the text of the comment when it executes a SHOW MODULE statement. Enclose the comment in single quotation marks (’) and separate multiple lines in a comment with a slash mark (/).

DETERMINISTIC
NOT DETERMINISTIC

These clauses are synonyms for the VARIANT and NOT VARIANT clauses for conformance to the SQL/PSM standard.

The DETERMINISTIC clause indicates that the same inputs to the function will generate the same output. It is the same as the NOT VARIANT clause.

The NOT DETERMINISTIC clause indicates that the output of the function does not depend on the inputs. It is the same as the VARIANT clause.

Usage Notes

- In general, SQL operates on modules, not stored routines, with the following exceptions: DROP FUNCTION, DROP PROCEDURE, SHOW FUNCTION, SHOW PROCEDURE, and CALL. The SHOW FUNCTION statement displays information about functions. The SHOW PROCEDURE statement displays individual procedures. The CALL statement can invoke only a single stored procedure.
- The mechanism-clause is not permitted for SQL stored functions or procedures.
- The following usage notes provide information about global variables:
  - Global variables can be referenced from any routine within the created module (just like local variables).
  - If a local variable and a global variable have the same name, then the local variable takes precedence over the global variable within the scope of the function or procedure in which the local variable is declared. For example:

```
SQL> SET FLAGS 'TRACE';
SQL> CREATE MODULE SAMPLE
cont>    DECLARE :IMAX INTEGER DEFAULT 100
cont>    PROCEDURE TRACE_MAX;
cont>    BEGIN
```

```
cont>    DECLARE :IMAX INTEGER DEFAULT 0;
cont>    TRACE :IMAX;
cont>    END;
cont> END MODULE;
SQL> CALL TRACE_MAX();
~Xt: 0
```

– All DEFAULT value clauses are evaluated when the first function or procedure is called for a module.

– Data persists until a module is unloaded at DISCONNECT time. Therefore, routines in a module can exchange data using global variables.

– Neither a COMMIT nor a ROLLBACK statement affects the state of the variables. That is, changes to variables are not rolled back.

Examples

Example 1 Adding Comments to Parameter Declarations

```
SQL> CREATE MODULE a
cont> LANGUAGE SQL
cont> PROCEDURE new_salary_proc
cont> (:id CHAR (5) COMMENT IS 'Employee ID',
cont> :new_salary INTEGER (2) COMMENT IS 'New Salary Amount');
cont> UPDATE SALARY_HISTORY
cont> SET SALARY_AMOUNT=:new_salary
cont> WHERE EMPLOYEE_ID=:id;
cont> END MODULE;
```

Example 2 Declaring a Global Variable to Exchange Information Between Two Routines

```
SQL> CREATE MODULE sample
cont>  LANGUAGE SQL
cont>  DECLARE :iter_count INTEGER
cont>  PROCEDURE set_iter (IN :val INTEGER)
cont>  COMMENT IS 'Validate the iteration count and assign'
cont>  /          'to a global variable.';
cont>  BEGIN
cont>  IF (:val IS NULL) OR (:val < 1) THEN
cont>     SIGNAL 'xxxxx'; --illegal value
cont>  ELSE
cont>     SET :iter_count  =:val;
cont>     TRACE 'Iteration count set to ', :val;
```

```
cont>  END IF;
cont>  END;
cont> FUNCTION GET_ITER ()
cont> RETURNS INTEGER
cont> COMMENT IS 'Trace the value used and then return the'
cont> / 'value from the global variable.';
cont> BEGIN
cont> TRACE 'Using iteration count ', :iter_count;
cont> RETURN :iter_count;
cont> END;
cont> END MODULE;
```

CREATE OUTLINE Statement

Creates a new query outline and stores this outline in the database.

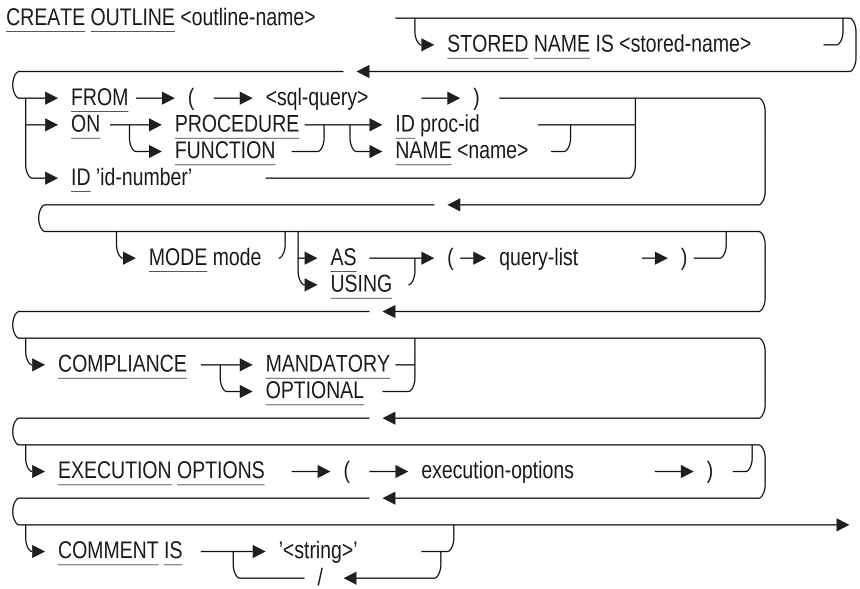
A query outline is an overall plan for how a query can be implemented. It may contain directives that control the join order, join methods, index usage (or all of these) that the optimizer selects when processing a query. Use of query outlines helps to ensure that query performance is highly stable across releases of Oracle Rdb.

Environment

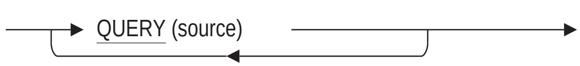
You can use the CREATE OUTLINE statement:

- In interactive SQL
- Embedded in host language programs to be precompiled
- As part of a procedure in an SQL module
- In dynamic SQL as a statement to be dynamically executed

Format



query-list =



source =

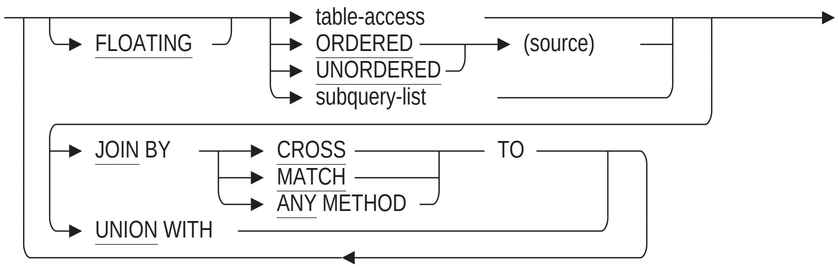
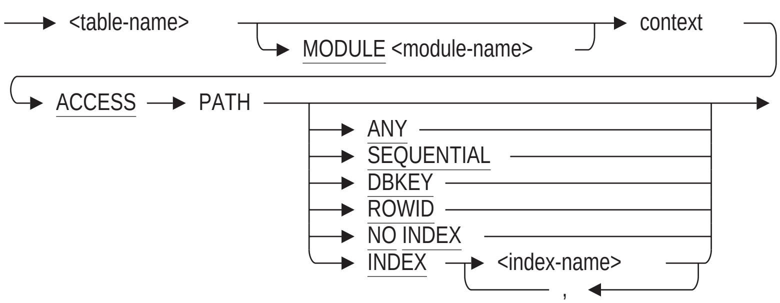
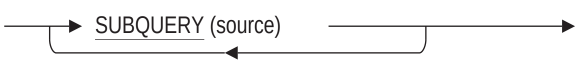


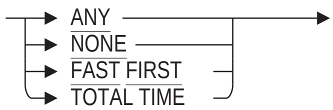
table-access =



subquery-list =



execution-options =



Arguments

No new arguments.

Usage Notes

- The order of the queries in an outline matches the order of optimization, not the order of execution. The query outline generated by Oracle Rdb appears with comments after the QUERY keyword in the outline to make reading easier. See the Examples section.
- The query outline generated by Oracle Rdb may not have a query corresponding to each statement within the procedure.
- Not all statements require the query optimizer. For example, TRACE and SET statements that do not reference tables do not require the optimizer.
- Subqueries in IF and CASE statements may be lifted into a previous statement by the optimizer to reduce the overhead associated with that query.
- Subqueries within an INSERT statement are executed as though SET statements were performed prior to the INSERT operation.
- INSERT statements are not subject to query optimization.
- During compilation of a constraint or trigger, Oracle Rdb will search for a query outline with the same name as the trigger or constraint being compiled. If a match is found, that outline will be used during the query compilation of the trigger or constraint. If no outline is found matching the name of the object, Oracle Rdb will then try to locate an appropriate outline using the BLR ID of the query.

  For example:

  .
  .
  .

```
SQL> CREATE TABLE TAB1 (a1 int CONSTRAINT TAB1NOTNULL NOT NULL ,
cont>                       a2 char(10),
cont>                       a3 char(10) );
SQL> CREATE OUTLINE TAB1NOTNULL
cont> id '8755644BCB040948E28A76B6D77CC2D3'
cont> MODE 0
cont> AS (
cont>   QUERY (
cont>     SUBQUERY (
cont>       TAB1 0  ACCESS PATH SEQUENTIAL
cont>       )
cont>     )
cont>   )
cont> COMPLIANCE OPTIONAL     ;
SQL> CREATE TRIGGER TAB1TRIG BEFORE INSERT ON TAB1
cont> (UPDATE TAB1 SET a3= 'bbbb' WHERE a2 = 'aaaa' ) FOR EACH ROW;
SQL> CREATE OUTLINE TAB1TRIG
cont> id '990F90B45658D27D64233D88D16AD273'
cont> MODE 0
cont> AS (
cont>   QUERY (
cont>     SUBQUERY (
cont>       TAB1 0  ACCESS PATH SEQUENTIAL
cont>       )
cont>     )
cont>   )
cont> COMPLIANCE OPTIONAL     ;
   .
   .
   .
$ DEFINE RDMS$DEBUG_FLAGS "SnsI"
   .
   .
   .
SQL> INSERT INTO tab1 (a1) VALUE (11);
~S: Trigger name  TAB1TRIG
~S: Outline TAB1TRIG used
~S: Outline TAB1NOTNULL used
Conjunct        Get     Retrieval sequentially of relation TAB1
   .
   .
   .
1 row inserted
SQL> commit;
~S: Constraint TAB1NOTNULL evaluated
```

```
Conjunct        Get     Retrieval sequentially of relation TAB1
   .
   .
   .
```

- If the TRACE statement is activated by the RDMS$DEBUG_FLAGS "Xt" logical name or by the SET FLAGS statement, queries in the TRACE statement are merged into the query outline for the procedure. For example, the following query outline contains one query when the TRACE statement is disabled:

```
SQL> DECLARE :ln CHAR(40);
SQL>
SQL> BEGIN
cont> TRACE 'Jobs Held: ',
cont>     (SELECT COUNT(*)
cont>        FROM job_history
cont>        WHERE employee_id = '00201');
cont> SELECT last_name
cont>     INTO :ln
cont>     FROM employees
cont>     WHERE employee_id = '00201';
cont> END;
-- Oracle Rdb Generated Outline : 28-MAY-1997 16:48
create outline QO_A17FA4B41EF1A68B_00000000
id 'A17FA4B41EF1A68B966C1A0B083BFDD4'
mode 0
as (
  query (
-- Select
    subquery (
      EMPLOYEES 0     access path index       EMPLOYEES_HASH
      )
    )
  )
compliance optional     ;
SQL>
```

  If the query outline is generated with TRACE enabled, two queries appear: the first is for the subquery in the TRACE statement and the second is for the singleton SELECT statement:

```
SQL> DECLARE :ln CHAR(40);
SQL>
SQL> BEGIN
cont> TRACE 'Jobs Held: ',
cont>     (SELECT COUNT(*)
```

```
cont>          FROM job_history
cont>          WHERE employee_id = '00201');
cont> SELECT last_name
cont>     INTO :ln
cont>     FROM employees
cont>     WHERE employee_id = '00201';
cont> END;
-- Oracle  Rdb Generated Outline : 28-MAY-1997 16:48
create outline QO_A17FA4B41EF1A68B_00000000
id 'A17FA4B41EF1A68B966C1A0B083BFDD4'
mode 0
as (
  query (
-- Trace
    subquery (
      JOB_HISTORY 0   access path index       JOB_HISTORY_HASH
      )
    )
  query (
-- Select
    subquery (
      EMPLOYEES 0     access path index       EMPLOYEES_HASH
      )
    )
  )
compliance optional     ;
~Xt: Jobs Held: 4
SQL>
```

If this second query outline is used at run time with the TRACE statement disabled, it cannot be applied to the query, as shown in the following example:

```
SQL> DECLARE :ln CHAR(40);
SQL>
SQL> BEGIN
cont> TRACE 'Jobs Held: ',
cont>     (SELECT COUNT(*)
cont>         FROM job_history
cont>         WHERE employee_id = '00201');
cont> SELECT last_name
cont>     INTO :ln
cont>     FROM employees
cont>     WHERE employee_id = '00201';
cont> END;
~S: Outline QO_A17FA4B41EF1A68B_00000000 used
```

```
~S: Outline/query mismatch; assuming JOB_HISTORY 0 renamed to EMPLOYEES 0
~S: Full compliance with the outline was not possible
Get     Retrieval by index of relation EMPLOYEES
  Index name  EMPLOYEES_HASH [1:1]       Direct lookup
```

Because the outline was created with compliance optional, the query outline is abandoned and a new strategy is calculated. If compliance is mandatory, the query fails.

If any TRACE statement contains a subquery, Oracle Corporation recommends using two query outlines (if any are required at all) with different modes in order to run the query with and without TRACE enabled. That is, when TRACE is enabled, define RDMS$BIND_OUTLINE_MODE to match the TRACE enabled query outlines.

```
$ DEFINE RDMS$DEBUG_FLAGS "Xt"
$ DEFINE RDMS$DEBUG_FLAGS_OUTPUT TRACE.DAT
$ DEFINE RDMS$BIND_OUTLINE_MODE 10
```

Alternatively, use the SET FLAGS statement, which allows the TRACE flag to be enabled and the MODE established from within an interactive session or through dynamic SQL. This scheme allows the query to be run with TRACE enabled or disabled.

- You can use the keyword MODE to set the query outline mode from within interactive and dynamic SQL session.

```
SQL> SET FLAGS 'MODE(10),OUTLINE';
SQL> SHOW FLAGS

Alias RDB$DBHANDLE:
Flags currently set for Oracle Rdb:
   PREFIX,OUTLINE,MODE(10)
SQL> SELECT COUNT(*) FROM employees;
-- Rdb Generated Outline : 30-MAY-1997 16:35
create outline QO_B3F54F772CC05435_0000000A
id 'B3F54F772CC054350B2B454D95537995'
mode 10 as (
  query (
-- For loop
    subquery (
      subquery (
        EMPLOYEES 0     access path index       EMP_EMPLOYEE_ID
        )
      )
```

```
    )
  )
compliance optional     ;

         100
1 row selected
```

The following options are accepted:

– NOMODE – this is the same as MODE(0) and disables the display of the mode in the SHOW FLAGS statement.

  A mode of zero is a valid mode setting and is the default for generated query outlines.

– MODE(*n*) – where *n* can be any numeric value (positive or negative).

– MODE – the same as MODE(1)

In the previous example, the mode was set to 10 when generating the query outline. If the generated outline is added to the database, it is used only when the mode is set to 10, either by the SET FLAGS statement or by using the logical name RDMS$BIND_OUTLINE_MODE.

- Consider the following procedure, which contains a FOR loop and an UPDATE statement nested within an outer FOR loop:

```
SQL> BEGIN
cont> -- Find the employee and
cont> -- complete their current job, before being promoted
cont> FOR :cur AS EACH ROW OF CURSOR a
cont>     FOR SELECT last_name
cont>         FROM EMPLOYEES
cont>         WHERE employee_id = :emp_id
cont> DO
cont>     BEGIN
cont>     -- Display some details
cont>     TRACE 'Employee: ', :cur.last_name;
cont>
cont>     FOR :cur2 AS EACH ROW OF CURSOR b
cont>         FOR SELECT cast(job_start AS DATE ANSI) AS js,
cont>                    cast(job_end AS DATE ANSI) AS je
cont>             FROM JOB_HISTORY
cont>             WHERE employee_id = :emp_id
cont>             ORDER BY job_start
cont>     DO
cont>         TRACE ' Job Duration: ',
```

```
cont>            (COALESCE (:cur2.je, current_date) - :cur2.js) YEAR TO
MONTH;
cont>     END FOR;
cont>
cont>     -- Now complete the current job
cont>     UPDATE JOB_HISTORY
cont>         SET job_end = CAST(current_date AS DATE VMS)
cont>         WHERE employee_id = :emp_id;
cont>
cont>     END;
cont> END FOR;
cont> END;
-- Oracle Rdb Generated Outline : 29-MAY-1997 22:52
create outline QO_39BBA6C4E902AB2B_00000000
id '39BBA6C4E902AB2B6A252A71A1CFFB71'
mode 0
as (
  query (
-- For loop
    subquery (
      EMPLOYEES 0     access path index       EMPLOYEES_HASH
      )
    )
  query (
-- For loop
    subquery (
      JOB_HISTORY 0   access path index       JOB_HISTORY_HASH
      )
    )
  query (
-- Update
    subquery (
      JOB_HISTORY 0   access path index       JOB_HISTORY_HASH
      )
    )
  )
compliance optional     ;
```

The order of the queries in the query outline represents a flattened tree structure that represents the complex execution profile of the compound statement. When extracting this tree structure, Oracle Rdb generates an order related to a bottom up representation of the optimization phase.

As a result, query outlines generated for any procedure with nested statements may appear inverted with the first table, EMPLOYEES, appearing last in the query outline.

```
-- Oracle Rdb Generated Outline : 29-MAY-1997 22:52
create outline QO_39BBA6C4E902AB2B_00000000
id '39BBA6C4E902AB2B6A252A71A1CFFB71'
mode 0
as (
  query (
-- For loop
    subquery (
      JOB_HISTORY 0   access path index       JOB_HISTORY_HASH
      )
    )
  query (
-- Update
    subquery (
      JOB_HISTORY 0   access path index       JOB_HISTORY_HASH
      )
    )
  query (
-- For loop
    subquery (
      EMPLOYEES 0     access path index       EMPLOYEES_HASH
      )
    )
  )
compliance optional     ;
```

Examples

Example 1 New Output from Query Outlines

```
SQL> BEGIN
cont> DECLARE :x INTEGER;
cont> -- Assignment
cont> SET :x = (SELECT COUNT(*) FROM TOUT_1);
cont> -- Delete statement
cont> DELETE FROM TOUT_1;
cont> -- Update statement
cont> UPDATE TOUT_1
cont>     SET a = (SELECT AVG(a) FROM TOUT_2)
cont>     WHERE a IS NULL;
```

```
cont> -- Singleton Select
cont> SELECT a INTO :x
cont>     FROM TOUT_1
cont>     WHERE a = 1;
cont> -- Trace (nothing if TRACE is disabled)
cont> TRACE 'The first value: ', (SELECT a FROM TOUT_1 LIMIT TO 1 ROW);
cont> END;
```

The query outline generated by Oracle Rdb appears with comments after the QUERY keyword in the outline.

```
-- Rdb Generated Outline : 29-MAY-1997 23:17
create outline QO_C11395E6020C6FFA_00000000
id 'C11395E6020C6FFA5A183A6CCE7C1F33'
mode 0
as (
  query (
-- Set
    subquery (
      TOUT_1 0        access path sequential
      )
    )
  query (
-- Delete
    subquery (
      TOUT_1 0        access path sequential
      )
    )
  query (
-- Update
    subquery (
      subquery (
        TOUT_2 1        access path sequential
        )
        join by cross to
      subquery (
        TOUT_1 0        access path sequential
        )
      )
    )
  query (
-- Select
    subquery (
      TOUT_1 0        access path sequential
      )
```

```
    )
  query (
-- Trace
    subquery (
      TOUT_1 0        access path sequential
      )
    )
  )
compliance optional     ;
```

CREATE PROFILE Statement

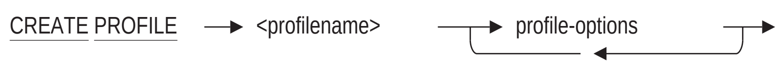
Creates a profile that extends a user definition within the database with special attributes that control transactions and resource usage.

Environment

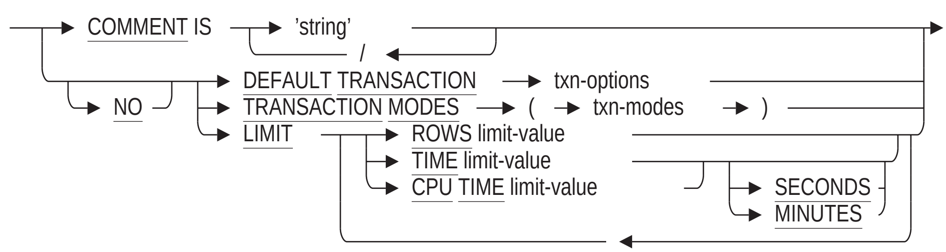
You can use the CREATE PROFILE statement:

- In interactive SQL
- Embedded in host language programs
- As part of a procedure in an SQL module
- In dynamic SQL as a statement to be dynamically executed

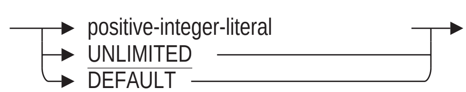
Format



profile-options =



limit-value =



Note: See the *Oracle Rdb7 SQL Reference Manual* for definitions of txn-options and txn-modes.

Arguments

COMMENT IS ’string’

This optional clause can be used to add several lines of comment to the profile object. The comment is displayed by the SHOW PROFILES statement. Enclose the comment in single quotation marks (’) and separate multiple lines in a comment with a slash mark (/).

DEFAULT TRANSACTION
NO DEFAULT TRANSACTION

DEFAULT TRANSACTION provides a default transaction for the user. By default, Oracle Rdb starts a READ WRITE transaction if none is explicitly started. If this definition is present for the current user, this definition will be used first. You can override this clause with a DECLARE or SET TRANSACTION statement.

NO DEFAULT TRANSACTION is the default.

Note: For this release, Oracle Rdb does not permit the RESERVING or EVALUATING clauses to appear in the default transaction.

TRANSACTION MODES
NO TRANSACTION MODES

TRANSACTION MODES provides the list of allowable transactions for this user. Please see the SET TRANSACTION MODES clause of the CREATE DATABASE and ALTER DATABASE statements for more details of txn-modes.

The transaction modes specified may include modes disabled for all database users by CREATE, IMPORT, or ALTER DATABASE statements. However, only the subset allowed by both profile and database settings will be used. For instance, if the database specifies (READ ONLY, SHARED READ, PROTECTED READ) and the profile specifies (READ ONLY, SHARED), the session will be allowed the subset (READ ONLY, SHARED READ).

NO TRANSACTION MODES is the default.

LIMIT ROWS
NO LIMIT ROWS
LIMIT ROWS sets the maximum number of rows that can be returned by a query started by the user.

NO LIMIT ROWS is the default.

Note: The run-time support for this clause is not complete for Oracle Rdb release 7.1.

LIMIT TIME
NO LIMIT TIME
LIMIT TIME sets the maximum elapsed time that can be used by the query compiler.

NO LIMIT TIME is the default.

Note: The run-time support for this clause is not complete for Oracle Rdb release 7.1.

LIMIT CPU TIME
NO LIMIT CPU TIME
LIMIT CPU TIME sets the maximum CPU time that can be used by the query compiler.

NO LIMIT CPU TIME is the default.

Note: The run-time support for this clause is not complete for Oracle Rdb release 7.1.

Examples

The following example specifies the allowed transaction modes for any user assigned this profile.

```
SQL> CREATE PROFILE DECISION_SUPPORT
cont>   COMMENT IS ’limit transactions used by report writers’
cont>   TRANSACTION MODES (NO READ WRITE, READ ONLY);
```

CREATE ROLE Statement

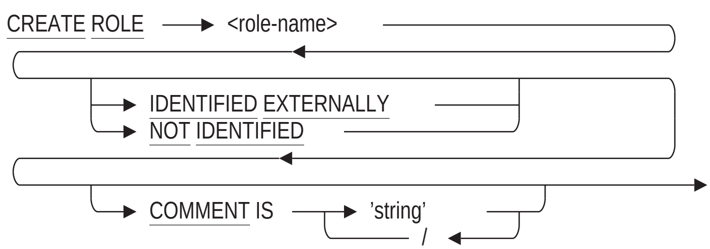
Creates a role to which privileges and other roles can be granted. A role can be granted to a user or another role. For example, you can create a role for members of a department. When a user leaves the department, the departmental role can be revoked from that user and thus exclude that user's access to the departmental files.

Environment

You can use the CREATE ROLE statement:

- In interactive SQL
- Embedded in host language programs to be precompiled
- As part of a procedure in an SQL module
- In dynamic SQL as a statement to be dynamically executed

Format



Arguments

role-name
A user-supplied name that you assign to the role. The special roles BATCH, DIALUP, INTERACTIVE, LOCAL, NETWORK, and REMOTE are reserved names that cannot be specified as a role-name. See the section on user-supplied names in the *Oracle Rdb7 SQL Reference Manual* for more information on valid user-supplied names.

IDENTIFIED EXTERNALLY
NOT IDENTIFIED

The IDENTIFIED EXERNALLY clause indicates that SQL should inherit the roles defined by the facilities of the operating system, such as rights identifiers. When a session is started, any role that is defined externally is established as part of the current user's profile.

The NOT IDENTIFIED clause indicates that SQL should not inherit any roles defined by the facilities of the operating system. This is the default.

COMMENT IS 'string'

Adds a comment about the role. SQL displays the text of the comment when it executes a SHOW ROLES statement. Enclose the comment in single quotation marks (') and separate multiple lines in a comment with a slash mark (/).

Usage Notes

- You must have the SECURITY privilege on the database to create a role.
- The special roles BATCH, DIALUP, INTERACTIVE, LOCAL, NETWORK, and REMOTE are granted by the OpenVMS operating system when the user process is created. Therefore, these roles are reserved names and cannot be used as the role-name in the CREATE ROLE statement.
- Oracle Rdb rightcases all user and role names that are identified externally. **Rightcasing** means that the names are stored as they are specified in the operating system registry or authorization database. On OpenVMS, all names are converted to uppercase, even if you use delimiters.

Examples

Example 1 Creating a Role

```
SQL> ALTER DATABASE FILENAME 'mf_personnel.rdb'
cont> SECURITY CHECKING IS INTERNAL;
SQL> ATTACH 'FILENAME mf_personnel.rdb';
SQL> CREATE ROLE WRITER;
SQL> SHOW ROLES;
Roles in database with filename mf_personnel.rdb
     WRITER
```

Example 2 Creating Roles and Granting Privileges to Those Roles

```
SQL> ALTER DATABASE FILENAME mf_personnel.rdb
cont> SECURITY CHECKING IS INTERNAL;
```

```
SQL> -- Create a role for employees in the payroll department
SQL> ATTACH 'FILENAME MF_PERSONNEL.RDB';
SQL> CREATE ROLE PAYROLL
cont> COMMENT IS 'This role allows access to various tables'
cont> /          'and procedures for use by the PAYROLL dept.';
SQL> -- Create another role for a subset of employees.
SQL> CREATE ROLE ANNUAL_LEAVE
cont> COMMENT IS 'This role is granted to PAYROLL personnel'
cont> /          'who adjust the annual leave data';
SQL> -- Grant EXECUTE privilege on module and ALL privilege on table
SQL> -- SALARY_HISTORY to all employees to whom the PAYROLL role has
SQL> -- been granted.  Grant EXECUTE privilege on module LEAVE_ADJUSTMENT
SQL> --  only to those employees who have been granted both the PAYROLL
SQL> -- and ANNUAL_LEAVE roles.
SQL> GRANT EXECUTE ON MODULE PAYROLL_UTILITIES TO PAYROLL;
SQL> GRANT ALL ON TABLE SALARY_HISTORY TO PAYROLL;
SQL> GRANT EXECUTE ON MODULE LEAVE_ADJUSTMENT
cont> to PAYROLL, ANNUAL_LEAVE;
SQL> -- User STUART joins the personnel department.  Grant him
SQL> -- the PAYROLL and ANNUAL_LEAVE roles so that he can
SQL> -- perform all functions in the payroll department.
SQL> CREATE USER STUART
cont> IDENTIFIED EXTERNALLY
SQL> GRANT PAYROLL, ANNUAL_LEAVE TO STUART;
SQL> -- User STUART is promoted to supervisor and thus
SQL> -- no longer needs access to the objects controlled by
SQL> -- the ANNUAL_LEAVE role.  Revoke that role from user
SQL> -- STUART.
SQL> REVOKE ANNUAL_LEAVE FROM STUART;
```

Create Routine Statement

Creates an external routine definition as a schema object in an Oracle Rdb database. **External routine** refers to both external functions and external procedures. A routine definition stores information in the database about a subprogram (a function or procedure) written in a 3GL language. The routine definition and the routine image are independent of each other, meaning one can exist without the other. However, to invoke an external routine, you need both the routine definition and routine image.

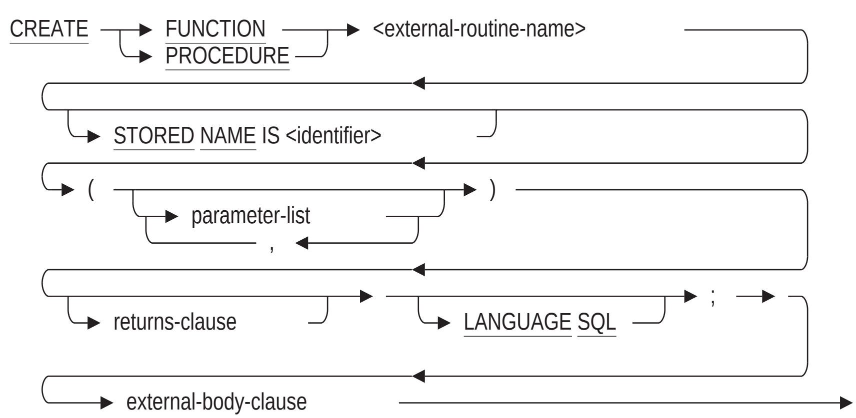
SQL can invoke an external function from anywhere you can specify a value expression. External procedures are invoked using the CALL statement for compound statements.

Environment

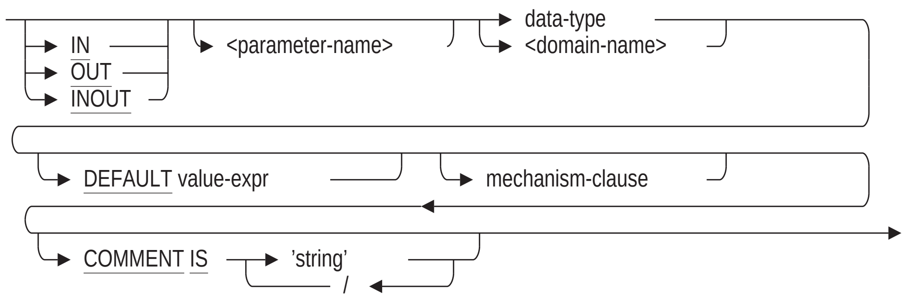
You can use the CREATE FUNCTION and CREATE PROCEDURE statements:

- In interactive SQL
- Embedded in host language programs to be precompiled
- As part of a procedure in an SQL module
- In dynamic SQL as a statement to be dynamically executed

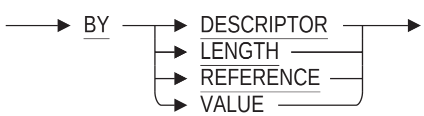
Format



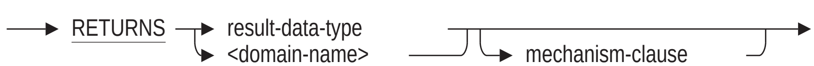
parameter-list =



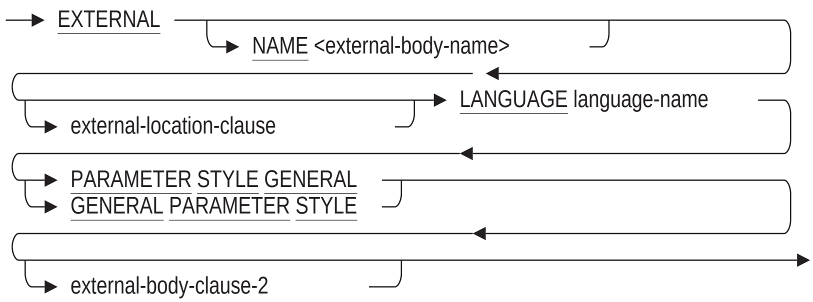
mechanism-clause =



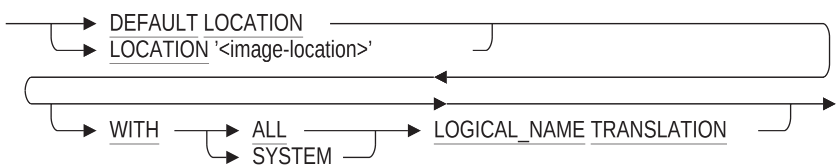
returns-clause =



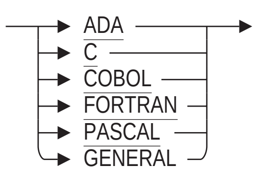
external-body-clause =



external-location-clause =

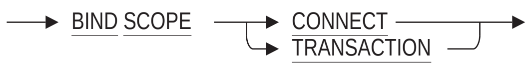


language-name =

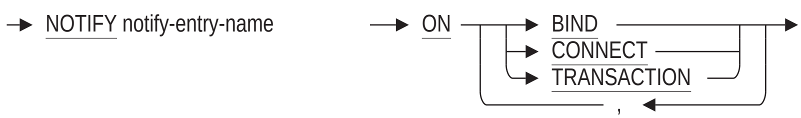


bind-site-clause =

bind-scope-clause =



notify-clause =



Arguments

DEFAULT value-expr
Specifies the default value of a parameter for an external function or procedure defined with mode IN. If you omit this parameter or if the Call statement argument list or function invocation specifies the DEFAULT keyword, then the default-value specified with this clause is used. The parameter uses NULL as the default if you do not specify a default value explicitly. However, this results in a run-time error unless you defined the external routine with the RETURNS NULL ON NULL INPUT clause.

COMMENT IS 'string'
Adds a comment about the function or procedure parameters. SQL displays the text of the comment when it executes a SHOW FUNCTIONS or SHOW PROCEDURES statement. Enclose the comment in single quotation marks (') and separate multiple lines in a comment with a slash mark (/).

PARAMETER STYLE GENERAL
Passes arguments and returns values by matching each parameter passed to an argument in the host language (3GL) routine. This clause is synonymous with the GENERAL PARAMETER STYLE clause. It is provided because this variant of the clause conforms to the SQL/PSM standard.

RETURNS NULL ON NULL INPUT
CALLED ON NULL INPUT

These clauses control how an external function is invoked when one or more of the function arguments is NULL. The CALLED ON NULL INPUT clause specifies that the function should be executed normally. A normal execution when the PARAMETER STYLE GENERAL clause is specified means that SQL should return a run-time error when the NULL value is detected.

The RETURNS NULL ON NULL INPUT clause instructs Oracle Rdb to avoid the function call and just return a NULL result. This option is valuable for library functions such as SIN, COS, CHECKSUM, SOUNDEX, and so on, that usually return an UNKNOWN result if an argument is NULL.

The CALLED ON NULL INPUT clause is the default.

DETERMINISTIC
NOT DETERMINISTIC

These clauses are synonyms for the VARIANT and NOT VARIANT clauses for conformance to the SQL/PSM standard.

The DETERMINISTIC clause indicates that the same inputs to the function will generate the same output. It is the same as the NOT VARIANT clause.

The NOT DETERMINISTIC clause indicates that the output of the function does not depend on the inputs. It is the same as the VARIANT clause.

Usage Notes

No new usage notes.

Examples

Example 1 Defining a Default Value for an IN Mode Parameter

```
SQL> CREATE PROCEDURE GIVE_RAISE
cont>   (IN :EMPLOYEE_ID CHAR(5)
cont>    COMMENT IS 'Target EMPLOYEE_ID',
cont>    IN :PERCENTAGE INTEGER(2) DEFAULT 1.1
cont>    COMMENT IS 'Percentage raise',
cont>    IN :EXTRA_BONUS INTEGER(2) DEFAULT 0);
cont> EXTERNAL
cont> LANGUAGE GENERAL
cont> PARAMETER STYLE GENERAL;
```

CREATE SEQUENCE Statement

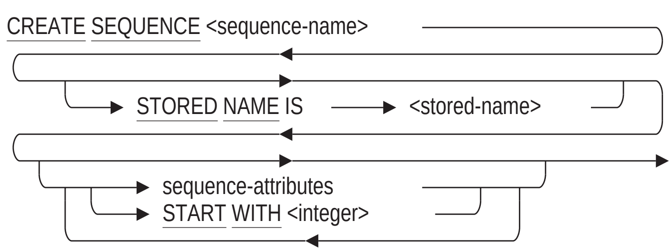
Creates a sequence. A sequence is a database object from which multiple users can generate unique integers. You can use sequences to automatically generate primary key values.

Environment

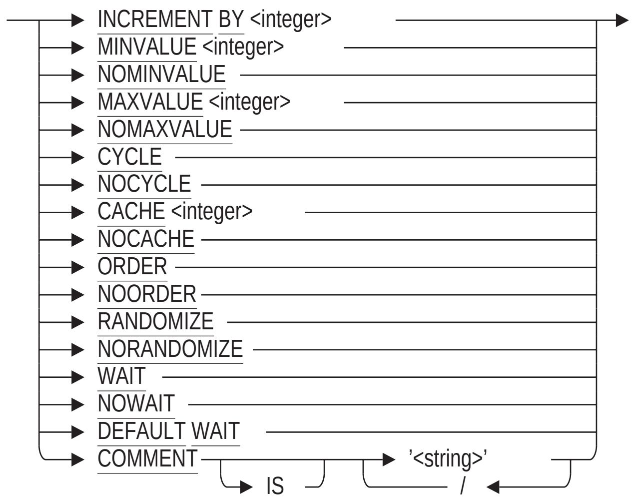
You can use the CREATE SEQUENCE statement:

- In interactive SQL
- Embedded in host language programs to be precompiled
- As part of a procedure in an SQL module
- In dynamic SQL as a statement to be dynamically executed

Format



sequence-attributes =



Arguments

sequence-name

The name of the sequence that you want to create. Use a name that is unique among all sequence names in the database, or in the schema if you are using a multischema database. Use any valid SQL name. (See the *Oracle Rdb7 SQL Reference Manual* for information on valid user-supplied names.)

STORED NAME IS stored-name

Specifies a name that Oracle Rdb uses to access a sequence created in a multischema database. The stored name allows you to access multischema definitions using interfaces, such as Oracle RMU, that do not recognize multiple schemas in one database. You cannot specify a stored name for a sequence in a database that does not allow multiple schemas. (See the *Oracle Rdb7 SQL Reference Manual* for more information on stored names.)

INCREMENT BY integer

Specifies the size of the increment and the direction (ascending or descending) of the sequence. This signed quadword (BIGINT) value must be in the range -2147483648 through 2147483647, excluding 0. The absolute value of the integer must be less than the difference of MAXVALUE and MINVALUE. A negative value specifies a descending sequence; a positive value specifies an ascending sequence. By default, the integer value is 1.

MINVALUE integer
NOMINVALUE

The MINVALUE clause specifies the minimum signed quadword (BIGINT) value that the sequence can generate. The integer value must be equal to or greater than -9223372036854775808. The MINVALUE must be less than or equal to the integer specified with the START WITH clause and less than the integer specified with the MAXVALUE clause. The NOMINVALUE clause specifies that the minimum value for an ascending sequence is 1, and -9223372036854775808 (plus the cache size) for a descending sequence.

MAXVALUE integer
NOMAXVALUE

The MAXVALUE clause specifies the maximum signed quadword (BIGINT) value that the sequence can generate. The integer value must be between -9223372036854775808 and 9223372036854775808. The MAXVALUE must be equal to or greater than the integer specified for the START WITH clause and greater than the integer specified with the MINVALUE clause. The NOMAXVALUE clause specifies that the maximum value for an ascending sequence is 9223372036854775808 (plus the cache size) and -1 for a descending sequence.

The NOMAXVALUE clause is the default.

CYCLE
NOCYCLE

The CYCLE clause specifies that the sequence is to continue generating values after reaching either the MINVALUE or MAXVALUE. After an ascending sequence reaches the MAXVALUE, the sequence starts again from its MINVALUE. After a descending sequence reaches its MINVALUE, the sequence starts again at its MAXVALUE. The NOCYCLE clause specifies that the sequence should not continue generating values after reaching either its minimum or maximum value. An error is generated if an attempt is made to increment the sequence beyond its limits. The NOCYCLE clause is the default.

CACHE integer
NOCACHE

The CACHE clause specifies how many values of the sequence Oracle Rdb should preallocate and keep in memory for faster access. The signed quadword (BIGINT) value must be between 2 and 2147483647.

You cannot cache more values than will fit in a given cycle of sequence numbers; thus, the maximum value allowed for the CACHE clause must be less than the value resulting from the following formula:

```
(MAXVALUE-MINVALUE)/ABS(INCREMENT).
```

A cache for a given sequence is populated at the first request for a number from that sequence. If a system failure occurs, all cached sequence values that have not been used in committed SQL statements are lost. The maximum number of lost values is equal to the integer value specified with the CACHE clause. The NOCACHE clause specifies that values of the sequence should not be preallocated.

By default, Oracle Rdb caches 20 sequence values.

ORDER
NOORDER

The ORDER clause specifies that sequence numbers are guaranteed to be assigned in order for each requesting process, thus maintaining a strict history of requests. The NOORDER clause specifies that sequence numbers are not guaranteed to be generated in order of request.

The NOORDER clause is the default.

RANDOMIZE
NORANDOMIZE

The RANDOMIZE clause specifies that the sequence numbers are to be returned with a random value in the most significant bytes of the BIGINT value. This allows unique values to be generated that have a random distribution. When you specify the NORANDOMIZE clause, sequence numbers are close in value to others created at the same time.

The advantage of the RANDOMIZE clause is that updates to columns of a sorted index to which these values are written occur in different locations in the index structure. This may improve concurrent access for large indexes that allow leaf nodes in different parts of the index to be updated independently. In contrast, the sequence numbers generated when you specify the NORANDOMIZE clause (which are likely to be close in numeric value to other sequences) result in index updates that occur in the same or nearby index nodes, which may lead to contention in one part of the sorted index.

The full range of values in the BIGINT value returned for the sequence are used; therefore, the NOMAXVALUE and NOMINVALUE clauses must be specified (or defaulted to) for the sequence definition. The most significant bits of the BIGINT value are set to a randomly generated positive value. A generated distinct value is returned in the least significant 32 bits so that uniqueness is guaranteed. If you also specify the CYCLE clause, then only the least significant 32 bits are cycled. When a query is performed on the column RDB$NEXT_SEQUENCE_VALUE in the RDB$SEQUENCES table, only the generated value of the least significant bits is returned, because the most significant bits are not assigned until the NEXTVAL pseudocolumn is referenced.

If you specify RANDOMIZE, you cannot specify ORDER, MAXVALUE, or MINVALUE. The NORANDOMIZE clause is the default.

WAIT
NOWAIT
DEFAULT WAIT

Specifies what wait state is used when a reference to NEXTVAL is used. A reference to NEXTVAL for a sequence may require synchronization with other users of the sequence. By default (or when you specify DEFAULT WAIT), the wait state (WAIT or NOWAIT) of the current transaction is used. This may mean that no waiting is performed during a NOWAIT transaction.

If you specify WAIT for the sequence, then regardless of the wait state set for the current transaction, all synchronization waits for the next value. This is the recommended setting if the application uses NOWAIT transactions. The current WAIT timeout interval defined for the transaction or database is used.

If you specify NOWAIT for the sequence, then regardless of the current transaction setting, all synchronization does not wait for the next value.

COMMENT IS ’string’

Adds a comment about the sequence. SQL displays the text of the comment when it executes a SHOW SEQUENCE statement. Enclose the comment in single quotation marks (’) and separate multiple lines in a comment with a slash mark (/).

START WITH integer

Specifies the initial signed quadword (BIGINT) value to be used for the sequence. This value must be in the range specified by (or defaulted to by) the other sequence attribute clauses. Valid values are in the range -9223373036854775808 to 9223372036854775807.

If omitted, the START WITH value defaults to the value of MINVALUE for ascending sequences and MAXVALUE for descending sequences.

Usage Notes

- You must have the CREATE database privilege on the database to create a sequence for that database.
- A user must have SELECT privileges on a sequence to use the NEXTVAL and CURRVAL pseudocolumns.
- A user must refer to the NEXTVAL pseudocolumn before he or she can use the CURRVAL pseudocolumn.
- A run-time lock is used to synchronize access to the next unused sequence value.
- Concurrent access is allowed to the sequence once the transaction in which the sequences were created is committed.
- If you specify the NEXTVAL pseudocolumn more than once in a statement, then only the first specification increments the sequence value; the others act as CURRVAL references.
- The value of the START WITH clause establishes the initial value generated after a sequence is created. This value is not necessarily the value to which an ascending cycling sequence cycles after reaching its maximum or minimum value.
- If you specify none of the sequence attributes, an ascending sequence is created that starts with 1, increases by 1, and has no upper limit. If the only sequence attribute that you specify is INCREMENT BY -1, a descending sequence that starts with -1 and decreases with no lower limit is generated.
- To create a sequence that increments without bounds, do one of the following:
  - For an ascending sequence, omit the MAXVALUE clause or specify the NOMAXVALUE clause.
  - For a descending sequence, omit the MINVALUE clause or specify the NOMINVALUE clause.
- To create a sequence that stops at a predefined limit, do one of the following:
  - For an ascending sequence, specify a value for the MAXVALUE clause and omit the CYCLE clause.
  - For a descending sequence, specify a value for the MINVALUE clause and omit the CYCLE clause. Any attempt to generate a sequence number once the sequence has reached its limit results in an error.

- To create a sequence that restarts after reaching a predefined limit, omit the CYCLE clause and specify values for both the MAXVALUE and MINVALUE clauses.
- Once a cache is created, you can access its values in SQL statements with the following pseudocolumns:
  - CURRVAL: Returns the current value of the sequence.
  - NEXTVAL: Increments the sequence and returns the new value.

Examples

Example 1 Creating a Sequence

```
SQL> -- This example creates a new sequence using the default
SQL> -- values for NOMINVALUE, NOMAXVALUE, INCREMENT BY 1, NOCYCLE,
SQL> -- and CACHE 20. The START WITH value is set to 147.
SQL> -- Allyn Stuart will be assigned an EMPLOYEE_ID value of 147.
SQL> -- Nick Jones will be assigned an EMPLOYEE_ID of 148.
SQL> --
SQL> CREATE SEQUENCE EMPID START WITH 00147;
SQL> -- Use NEXTVAL to fetch a sequence number for the primary key column.
SQL> INSERT INTO EMPLOYEES
cont> (EMPLOYEE_ID, LAST_NAME, FIRST_NAME)
cont> VALUES (EMPID.NEXTVAL, 'STUART', 'ALLYN')
cont> RETURNING EMPLOYEE_ID;
 EMPLOYEE_ID
 147
1 row inserted
SQL> -- Use CURRVAL to reuse the EMPLOYEE_ID value for the foreign key columns
SQL> -- in the associated tables.
SQL> INSERT INTO SALARY_HISTORY
cont> (EMPLOYEE_ID, SALARY_AMOUNT,SALARY_START, SALARY_END)
cont> VALUES (EMPID.CURRVAL, 35000, '6-FEB-1998', NULL)
cont> RETURNING EMPLOYEE_ID;
 EMPLOYEE_ID
 147
1 row inserted
SQL> INSERT INTO JOB_HISTORY
cont> (EMPLOYEE_ID, DEPARTMENT_CODE, JOB_START, JOB_END)
cont> VALUES (EMPID.CURRVAL, 'ENGR', '6-FEB-1998', NULL)
cont> RETURNING EMPLOYEE_ID;
 EMPLOYEE_ID
 147
1 row inserted
```

```
SQL> INSERT INTO EMPLOYEES
cont> (EMPLOYEE_ID, LAST_NAME, FIRST_NAME)
cont> VALUES (EMPID.NEXTVAL,'JONES','NICK')
cont> RETURNING EMPLOYEE_ID;
 EMPLOYEE_ID
 148
1 row inserted
```

CREATE STORAGE MAP Statement

Associates a table with one or more storage areas in a multifile database. The CREATE STORAGE MAP statement specifies a **storage map** that controls which lists or rows of a table are stored in which storage areas.

In addition to creating storage maps, the CREATE STORAGE MAP statement has options that control:

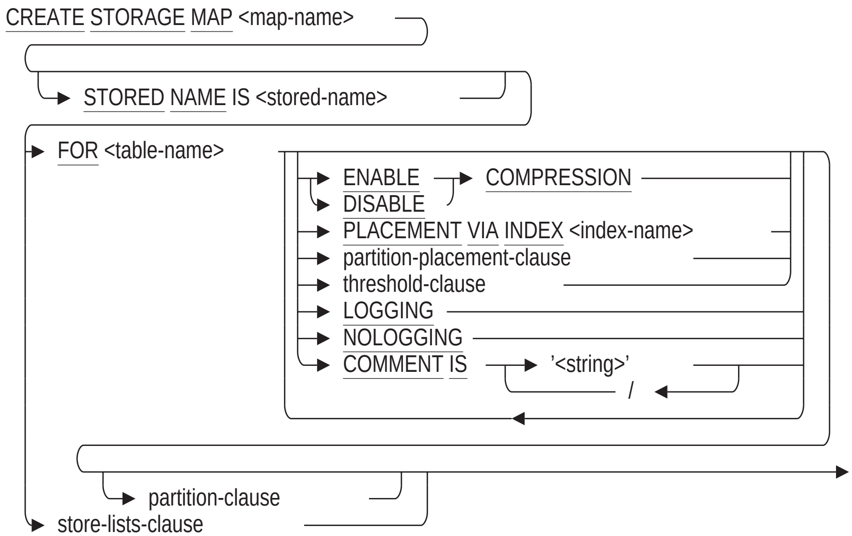
- Which index the database system uses when inserting rows in the table
- Whether or not the rows of the table are stored in a compressed format
- Whether or not partitioning keys can be modified
- Whether the table is partitioned vertically, horizontally, or both
- Whether logging to the .aij and .ruj files (logging) is enabled or disabled for the duration of this operation

Environment

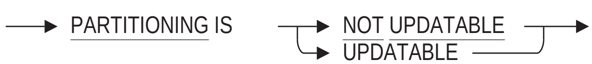
You can use the CREATE STORAGE MAP statement:

- In interactive SQL
- Embedded in host language programs to be precompiled
- As part of a procedure in an SQL module
- In dynamic SQL as a statement to be dynamically executed

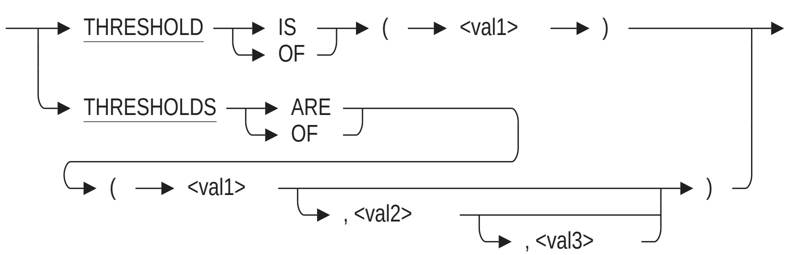
Format



partition-placement-clause =



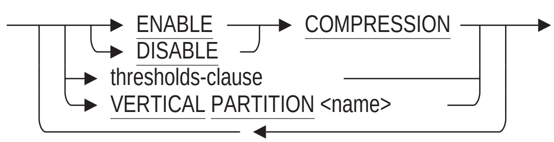
threshold-clause =



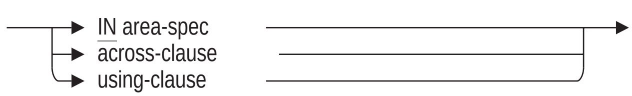
partition-clause =

columns-clause =

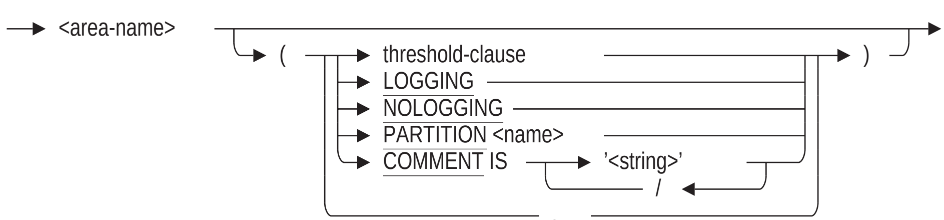
store-attributes =



store-clause =

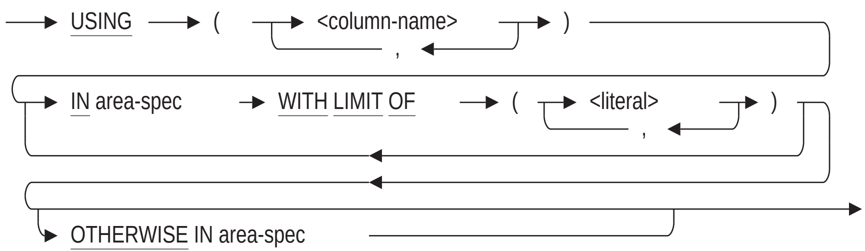


area-spec =

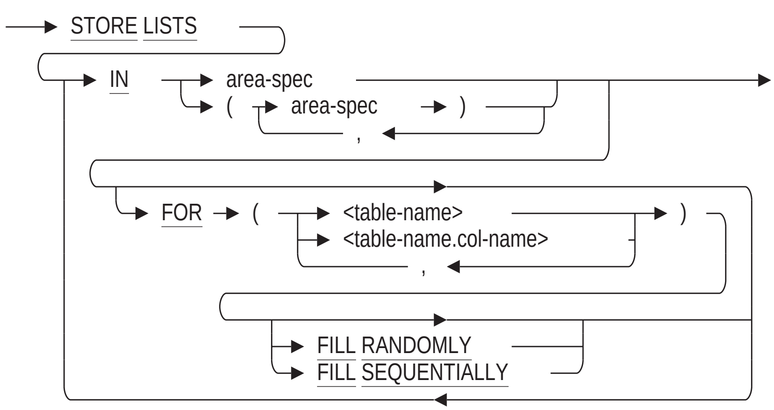


across-clause =

using-clause =



store-lists-clause =



Arguments

LOGGING
NOLOGGING
The LOGGING clause specifies that the CREATE STORAGE MAP statement should be logged in the recovery-unit journal file (.ruj) and after-image journal file (.aij).

The NOLOGGING clause specifies that the CREATE STORAGE MAP statement should not be logged in the recovery-unit journal file (.ruj) and after-image journal file (.aij).

The LOGGING clause is the default.

See Section 2.2.8 for information on the advantages and disadvantages of specifying these clauses and the implications that using the NOLOGGING keyword has on database recovery.

PARTITION name
Names the partition. The name can be a delimited identifier if the dialect or quoting rules are set to SQL92 or SQL99. Partition names must be unique within the storage map. If you do not specify this clause, Oracle Rdb generates a default name for the partition.

COMMENT IS 'string'
Adds a comment about the storage map. SQL displays the text of the comment when it executes a SHOW STORAGE MAPS statement. Enclose the comment in single quotation marks (') and separate multiple lines in a comment with a slash mark (/).

VERTICAL PARTITION name
Names a vertical partition. The name can be a delimited identifier if the dialect or quoting rules are set to SQL92 or SQL99. Partition names must be unique within the storage map. If you do not specify this clause, Oracle Rdb generates a default name for the partition.

Usage Notes

- You can now successfully issue a CREATE STORAGE MAP statement even if a table contains data. However, no storage map (other than the default) can currently exist for this table.

  The storage map that you create must be a simple map that references only the default storage area and represents the current (default) mapping for the table.

The default storage area is either RDB$SYSTEM or the area name provided by the CREATE DATABASE DEFAULT STORAGE AREA clause.

The new map cannot change thresholds or compression for the table, nor can it use the PLACEMENT VIA INDEX clause or the WITH LIMIT clause. It can contain only one area and cannot be vertically partitioned. This new map simply describes the mapping as it exists by default for the table. If you attempt to use the WITH LIMIT clause under these circumstances, the following messages are returned:

```
%RDB-E-NO_META_UPDATE, metadata update failed
-RDMS-F-RELNOTEMPTY, table <table-name> has data in it
-RDMS-E-NOCMPLXMAP, can not use complex map for non-empty table
```

Once the storage map is created, you can use the ALTER STORAGE MAP clause to reorganize the table as required. This is shown in Example 2 in the Examples section.

If the new storage map contains any unacceptable attributes it will be rejected, as shown in Example 3 in the Examples section.

- Some system tables are automatically created in the secondary system area if defined by the clause DEFAULT STORAGE AREA in the CREATE DATABASE statement. Additionally, a set of optional system tables exists (which may not exist in all databases) that can be mapped manually to other storage areas.

  The set of system tables for which you can change the mapping, and the instructions on how to do so, are provided in the section on moving certain system tables to separate storage areas in the *Oracle Rdb7 Guide to Database Design and Definition*.

Examples

Example 1 Disabling Logging and Naming Horizontal and Vertical Partitions

```
SQL> CREATE DATABASE FILENAME birdlist
cont>     CREATE STORAGE AREA AREA1
cont>     CREATE STORAGE AREA AREA2
cont>     CREATE STORAGE AREA AREA3
cont>     CREATE STORAGE AREA AREA4
cont>     CREATE STORAGE AREA AREA5
cont>     CREATE STORAGE AREA AREA6
cont>     CREATE STORAGE AREA AREA7
cont>     CREATE STORAGE AREA AREA8
cont>     CREATE TABLE SPECIES
```

```
cont>   ( GENUS          CHAR (30),
cont>     SPECIES        CHAR (30),
cont>     COMMON_NAME    CHAR (40),
cont>     FAMILY_NUMBER  INT (3),
cont>     SPECIES_NUMBER INT (3)
cont>   )
cont>     CREATE INDEX I1 ON SPECIES (FAMILY_NUMBER)
cont>     CREATE TABLE SIGHTING
cont>   ( SPECIES_NUMBER INT (3),
cont>     COMMON_NAME    CHAR (40),
cont>     CITY  CHAR (20),
cont>     STATE CHAR (20),
cont>     SIGHTING_DATE  DATE ANSI,
cont>     NOTES_NUMBER INT (5))
cont>     CREATE INDEX I2 ON SIGHTING (SPECIES_NUMBER)
cont>     CREATE TABLE FIELD_NOTES
cont>   ( WEATHER CHAR (30),
cont>     TIDE CHAR (15),
cont>     SIGHTING_TIME TIMESTAMP(2),
cont>     NOTES CHAR (500),
cont>     NOTES_NUMBER INT (5),
cont>     SPECIES_NUMBER INT (3))
cont>     CREATE INDEX I3 ON FIELD_NOTES (NOTES_NUMBER);
SQL> --
SQL> -- Note that the default threshold clause for the
SQL> -- storage map is not enclosed in parentheses, but each
SQL> -- threshold clause associated with a particular area is enclosed
SQL> -- in parentheses.
SQL> --
SQL> CREATE STORAGE MAP M1 FOR SPECIES
cont>     THRESHOLDS ARE (30, 50, 80)
cont>     ENABLE COMPRESSION
cont>     PLACEMENT VIA INDEX I1
cont>     NOLOGGING
cont>     COMMENT IS 'Storage Map for Species'
cont>     STORE
cont>   IN AREA1
cont>       (THRESHOLD (10),
cont>        PARTITION AREA1,
cont>        COMMENT IS 'Partition is AREA1');
SQL> --
SQL> CREATE STORAGE MAP M2 FOR SIGHTING
cont>   THRESHOLD IS (40)
cont>     STORE
cont>   RANDOMLY ACROSS (
```

```
cont>       AREA1 (THRESHOLD OF (10),
cont>       PARTITION AREA1),
cont>       AREA2 (THRESHOLDS ARE (30, 50, 98),
cont>       PARTITION AREA2),
cont>       AREA3 (PARTITION AREA3)
cont>   );
SQL> --
SQL> CREATE STORAGE MAP M3 FOR FIELD_NOTES
cont>   THRESHOLDS OF (50,70,90)
cont>     STORE COLUMNS (WEATHER, TIDE, SIGHTING_TIME)
cont>     VERTICAL PARTITION WEATHER_TIDE_SIGHTINGTIME
cont>   USING (SPECIES_NUMBER, NOTES_NUMBER)
cont>       IN AREA1
cont>           (THRESHOLDS OF (20, 80, 90) )
cont>           WITH LIMIT OF (30, 88)
cont>       IN AREA2
cont>           WITH LIMIT OF (40, 89)
cont>       IN AREA3
cont>           WITH LIMIT OF (50, 90)
cont>       OTHERWISE IN AREA4
cont>           (THRESHOLDS ARE (20, 30, 40))
cont>   STORE COLUMNS (NOTES, NOTES_NUMBER, SPECIES_NUMBER)
cont>    VERTICAL PARTITION NOTES_NOTESNUM_SPECIESNUM
cont>    USING (SPECIES_NUMBER)
cont>       IN AREA5
cont>           (THRESHOLDS OF (20, 80, 90) )
cont>           WITH LIMIT OF (30)
cont>       IN AREA6
cont>           WITH LIMIT OF (40)
cont>       IN AREA7
cont>           WITH LIMIT OF (50)
cont>       OTHERWISE IN AREA8
cont>           (THRESHOLDS ARE (20, 30, 40));
```

Example 2 Creating a Storage Map for a Table Containing Data

```
SQL> -- Create table, insert data, and then create a storage map.
SQL> --
SQL> CREATE TABLE MAP_TEST2 (a INTEGER, b CHAR(10));
SQL> INSERT INTO MAP_TEST2 (a, b) VALUES (2, 'Second');
1 row inserted
SQL> CREATE STORAGE MAP MAP_TEST2_MAP FOR MAP_TEST2
cont>     STORE IN RDB$SYSTEM;
SQL> INSERT INTO MAP_TEST2 (a, b) VALUES (22, 'Second2');
1 row inserted
```

```
SQL> COMMIT;
SQL> SELECT *,DBKEY FROM MAP_TEST2;
           A   B                               DBKEY
           2   Second                       90:809:0
          22   Second2                      90:809:1
2 rows selected
SQL>
SQL> -- Now alter the storage map and
SQL> -- place it in a different storage area.
SQL>
SQL> ALTER STORAGE MAP MAP_TEST2_MAP
cont>     STORE IN TEST_AREA2;
SQL> COMMIT;
SQL> SELECT *,DBKEY FROM MAP_TEST2;
           A   B                               DBKEY
           2   Second                        91:11:0
          22   Second2                       91:11:1
2 rows selected
SQL>
```

Example 3 Invalid Attempts to Create a Storage Map

```
SQL> -- Create table, insert data, and then
SQL> -- create a storage map with invalid attributes.
SQL>
SQL> CREATE TABLE MAP_TEST3 (a INTEGER, b CHAR(10));
SQL> CREATE INDEX MAP_TEST3_INDEX ON MAP_TEST3 (a);
SQL> INSERT INTO MAP_TEST3 (a, b) VALUES (3, 'Third');
1 row inserted
SQL>
SQL> CREATE STORAGE MAP MAP_TEST3_MAP FOR MAP_TEST3
cont>     STORE IN TEST_AREA1;            -- Must be the default area.
%RDB-E-NO_META_UPDATE, metadata update failed
-RDMS-F-RELNOTEMPTY, table "MAP_TEST3" has data in it
-RDMS-E-NOCMPLXMAP, can not use complex map for non-empty table
SQL>
SQL> CREATE STORAGE MAP MAP_TEST3_MAP for MAP_TEST3
cont>     PLACEMENT VIA INDEX MAP_TEST3_INDEX   -- Can't use placement.
cont>     STORE IN RDB$SYSTEM;
%RDB-E-NO_META_UPDATE, metadata update failed
-RDMS-F-RELNOTEMPTY, table "MAP_TEST3" has data in it
-RDMS-E-NOCMPLXMAP, can not use complex map for non-empty table
SQL>
SQL> CREATE STORAGE MAP MAP_TEST3_MAP FOR MAP_TEST3
cont>     DISABLE COMPRESSION              -- Can't change compression.
```

```
cont>     STORE IN RDB$SYSTEM;
%RDB-E-NO_META_UPDATE, metadata update failed
-RDMS-F-RELNOTEMPTY, table "MAP_TEST3" has data in it
-RDMS-E-NOCMPLXMAP, can not use complex map for non-empty table
SQL>
SQL> CREATE STORAGE MAP MAP_TEST3_MAP for MAP_TEST3
cont>     THRESHOLDS ARE (50, 60, 70)        -- Can't change thresholds.
cont>     STORE IN RDB$SYSTEM;
%RDB-E-NO_META_UPDATE, metadata update failed
-RDMS-F-RELNOTEMPTY, table "MAP_TEST3" has data in it
-RDMS-E-NOCMPLXMAP, can not use complex map for non-empty table
SQL>
SQL> CREATE STORAGE MAP MAP_TEST3_MAP FOR MAP_TEST3
cont>     STORE ACROSS (RDB$SYSTEM, TEST_AREA2);-- Can't use more than one area.
%RDB-E-NO_META_UPDATE, metadata update failed
-RDMS-F-RELNOTEMPTY, table "MAP_TEST3" has data in it
-RDMS-E-NOCMPLXMAP, can not use complex map for non-empty table
SQL>
SQL> CREATE STORAGE MAP MAP_TEST3_MAP for MAP_TEST3
cont>     STORE COLUMNS (a) in RDB$SYSTEM       -- Can't vertically partition.
cont>     STORE COLUMNS (b) in TEST_AREA2;
%RDB-E-NO_META_UPDATE, metadata update failed
-RDMS-F-RELNOTEMPTY, table "MAP_TEST3" has data in it
-RDMS-E-NOCMPLXMAP, can not use complex map for non-empty table
```

CREATE SYNONYM Statement

Creates an alternate name or synonym for an existing database object. The object may be a domain, function, module, procedure, sequence, another synonym, table, or view.

Once defined, the synonym can be used in any query or data definition language statement in place of the referenced object.

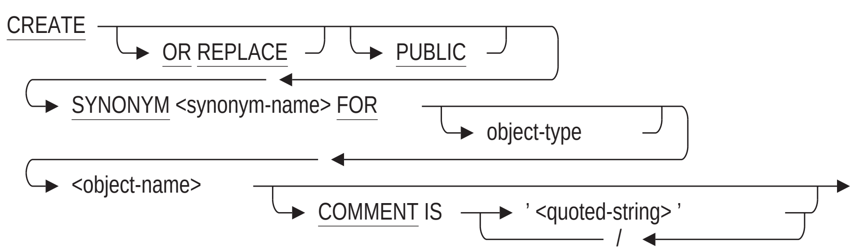
However, the SHOW commands do not accept synonyms. Use the SHOW SYNONYM statement to see if the name is a synonym.

Environment

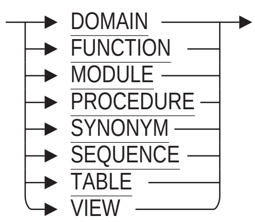
You can use the CREATE SYNONYM or CREATE OR REPLACE SYNONYM statement:

- In interactive SQL
- Embedded in host language programs
- As part of a procedure in an SQL module
- In dynamic SQL as a statement to be dynamically executed

Format



object-type =



Arguments

OR REPLACE

This clause instructs SQL to replace any synonym of this name if it exists. If it does not exist, a new synonym is created. This shorthand allows replacement of an existing synonym while maintaining all the dependencies established by query and DDL usage of this synonym.

PUBLIC

This optional clause is provided for compatibility with the Oracle database server. It is currently not used by Oracle Rdb. Its presence or absence may be used by future releases. Oracle Corporation recommends you use the PUBLIC keyword in applications.

synonym-name

The name of the synonym you want to create. The synonym name must be unique within all domains, tables, views, functions, procedures, modules, sequences, and synonyms within the database. You may qualify it with an alias.

DOMAIN
FUNCTION
MODULE
PROCEDURE
SEQUENCE
SYNONYM
TABLE
VIEW

These optional object types can be used when the referenced object name is not unique within the database. For instance, Oracle Rdb allows a domain and a table to both be called MONEY. Therefore, to create a synonym for the table MONEY, you must use the FOR TABLE clause so that it is uniquely identified.

FOR object-name

The name of the database object for which the synonym is required. This name must exist for an object in the database. If the optional object type is omitted, then Oracle Rdb will search the database for an object with this name.

COMMENT IS ’quoted-string’

This optional clause can be used to add several lines of comment to the synonym object. The comment is displayed by the SHOW SYNONYM statement.

Usage Notes

- The database must be altered to enable synonym creation. The ALTER DATABASE ... SYNONYMS ARE ENABLED clause creates a new system relation, RDB$OBJECT_SYNONYMS, which is used to record the synonyms created by this statement.
- Synonyms do not have any access control. Instead, granting privileges to, or revoking privileges from a synonym is the same as referencing the base object. In the following example, the GRANT statement grants the SELECT privilege to PUBLIC on the EMPLOYEES table:

  ```
  SQL> CREATE SYNONYM EMPS FOR EMPLOYEES;
  SQL> GRANT SELECT ON TABLE EMPS TO PUBLIC;
  ```

- You must have the database CREATE privilege to execute the CREATE SYNONYM statement.
- You must have the REFERENCES privilege on the referenced object to create a synonym for that object. Because domains do not have access control, no other privileges are required to create synonyms for domains.
- You may create synonyms for synonyms. This forms a chain of synonyms that must be processed to determine the base database object. Oracle Corporation recommends that this chain be no more than 10 references. Oracle Rdb enforces a chain maximum length of 64.

Examples

Example 1 Using the Default Alias

```
SQL> CREATE SYNONYM emps FOR employees;
```

Example 2 Using an Explicit Alias for the Synonym

```
SQL> CREATE SYNONYM db1.emps FOR employees;
```

Example 3 Using an Explicit Alias for the Referenced Object

```
SQL> CREATE SYNONYM emps FOR db1.employees;
```

Example 4 Using the Alias Explicitly

```
SQL> CREATE SYNONYM db1.emps FOR db1.employees;
```

Example 5 Using the Table Type

```
SQL> CREATE SYNONYM cash FOR table money
cont>   COMMENT IS 'use a different name to avoid confusion with'
cont>   /          'the domain MONEY';
```

Example 6 Using Multiple Synonyms

```
SQL> CREATE TABLE t_employees_0001 (...);
SQL> CREATE SYNONYM employees FOR t_employees_0001;
SQL> CREATE SYNONYM emps FOR employees;
```

CREATE TABLE Statement

Creates a temporary or persistent base table definition or an information table. A table definition consists of a list of definitions of columns that make up a row in the table.

Persistent base tables are tables whose metadata and data are stored in the database beyond an SQL session. The data can be shared by all users attached to the database.

Temporary tables are tables whose data is automatically deleted when an SQL session or module ends. The tables materialize only when you refer to them in an SQL session, and the data does not persist beyond an SQL session. You can also specify whether the data is preserved or deleted at the end of a transaction within the session; the default is to delete the data. The data in temporary tables is private to the user. There are three types of temporary tables:

- Global temporary tables
- Local temporary tables
- Declared local temporary tables (see the entry for the DECLARE LOCAL TEMPORARY TABLE Statement in the *Oracle Rdb7 SQL Reference Manual* for additional information)

The metadata for a global temporary table is stored in the database and persists beyond the SQL session. Different SQL sessions can share the same metadata. The data stored in the table cannot be shared between SQL sessions. However, the data can be shared between modules in a single SQL session. The data does not persist beyond an SQL session.

The metadata for a local temporary table is stored in the database and persists beyond the SQL session. Different SQL sessions can share the same metadata. The data stored in the table cannot be shared between different modules in a single SQL session or between SQL sessions. The data does not persist beyond an SQL session or module.

Because temporary tables are used only to hold the user's data, which is not shared among users, no locks are needed and the data can be modified in a read-only transaction.

See the *Oracle Rdb7 Guide to Database Design and Definition* for more information on temporary tables.

Information tables are special read-only tables that can be created in an Oracle Rdb release 7.1 database and used to retrieve database attributes that are not stored in the existing relational tables. Information tables allow interesting database information, which is currently stored in an internal format, to be displayed as a relational table. See Section 2.1 for more information.

When you define a table, you can also define table constraints. A **constraint** specifies a condition that restricts the values that can be stored in a table. Constraints can specify that when you define a table, you can also define table constraints. A **constraint** specifies a condition that restricts the values that can be stored in a table. Constraints can specify that columns contain:

- Only certain values
- Primary key values
- Unique values
- Values that cannot be null

There are two ways to specify a table definition in the CREATE TABLE statement:

- Directly, by naming the table, its columns and associated data types, default values (optional), constraint definitions (optional), and formatting clauses

  You can define constraints on persistent base tables and global temporary tables only.
- Indirectly, by providing a path name for a repository record definition that specifies the table name, columns, and data types

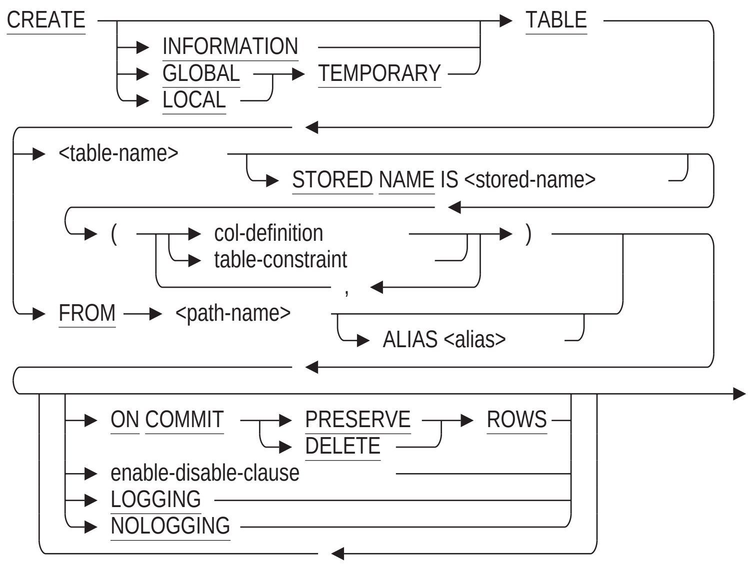
SQL allows you to specify the default character data type or the national character data type when defining table columns.

Environment

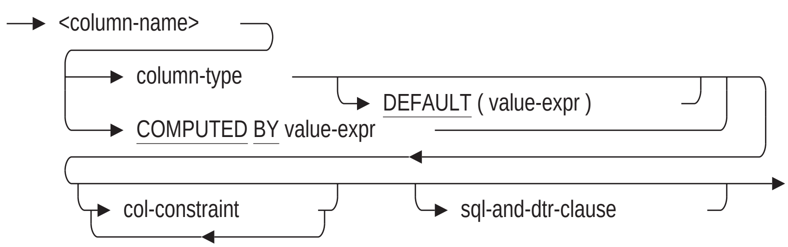
You can use the CREATE TABLE statement:

- In interactive SQL
- Embedded in host language programs to be precompiled
- As part of a procedure in an SQL module
- In dynamic SQL as a statement to be dynamically executed

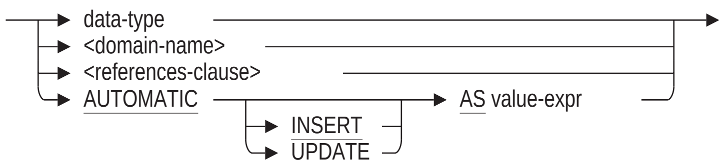
Format



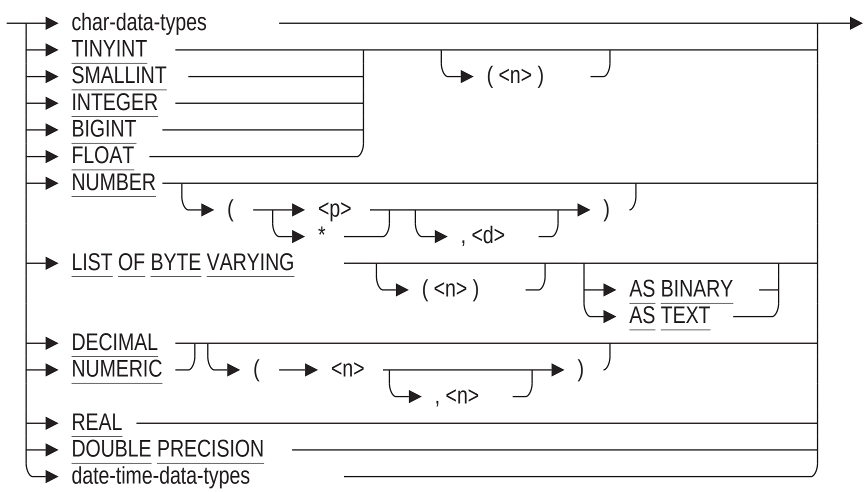
col-definition =



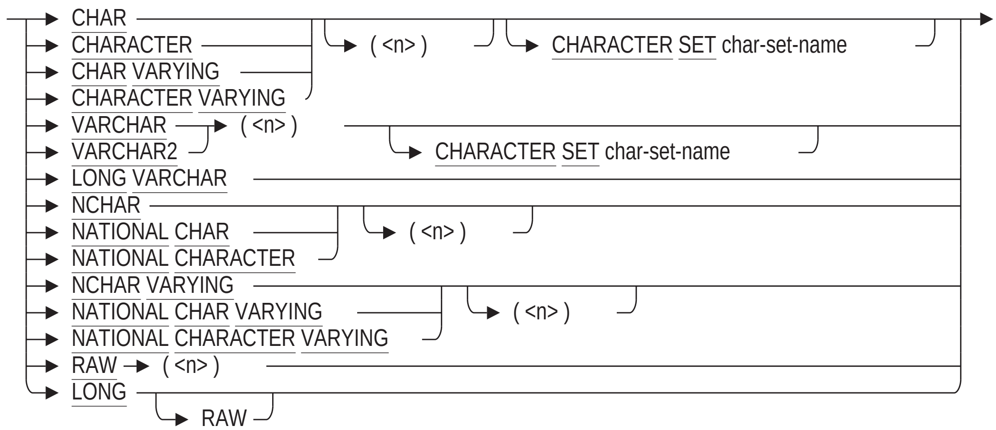
column-type=



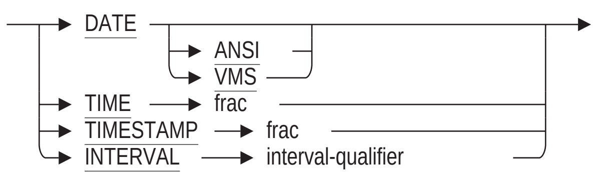
data-type =



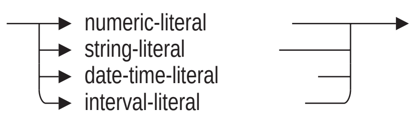
char-data-types =



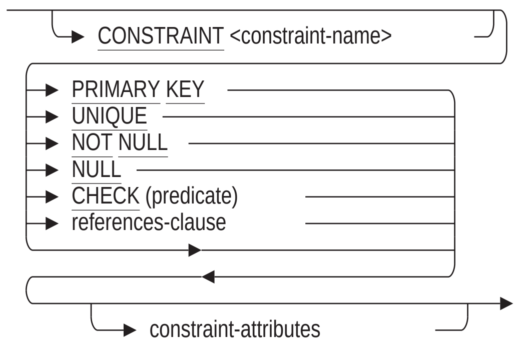
date-time-data-types =



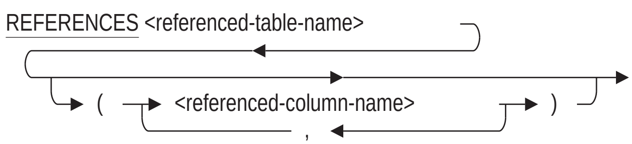
literal =



col-constraint=



references-clause =



sql-and-dtr-clause =

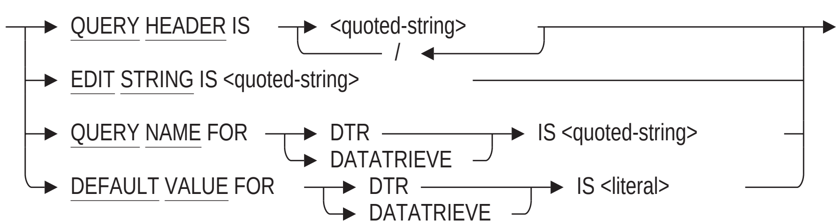


table-constraint =

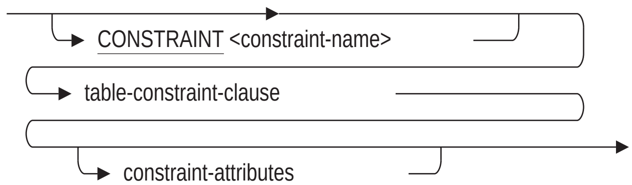
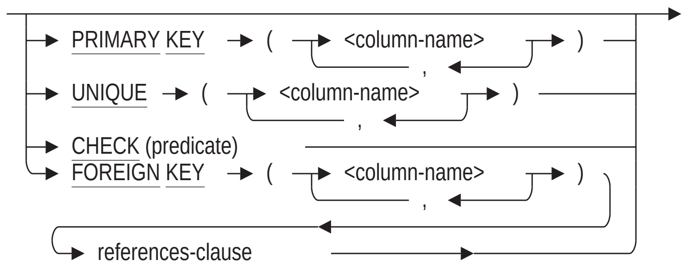
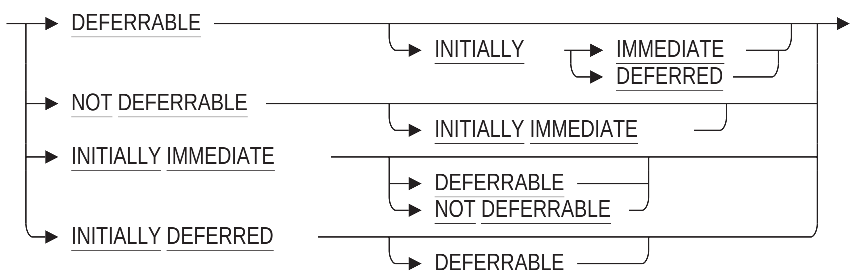


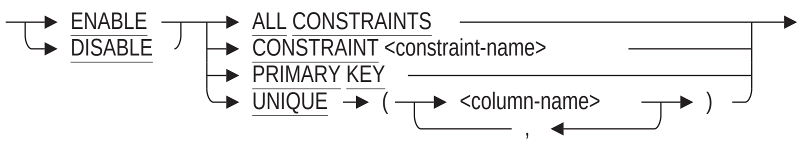
table-constraint-clause =



constraint-attributes =



enable-disable-clause =



Arguments

CREATE INFORMATION TABLE

Specifies that the table definition is an information table. For details on information tables, see Section 2.1.

Information tables are reserved for use by Oracle Corporation.

AUTOMATIC AS value-expr
AUTOMATIC INSERT AS value-expr
AUTOMATIC UPDATE AS value-expr

These AUTOMATIC column clauses allow you to store special information when data is inserted into a row or a row is updated. For example, you can log application-specific information to audit activity or provide essential values, such as time stamps or unique identifiers for the data.

The assignment of values to these types of columns is managed by Oracle Rdb. The AUTOMATIC INSERT clause can be used to provide a complex default for the column when the row is inserted; it cannot be changed by an UPDATE statement. The AUTOMATIC UPDATE clause can be used to provide an updated value during an UPDATE statement. The AUTOMATIC clause is the default and specifies that the value expression should be applied during both INSERT and UPDATE statements. The column type is derived from the AS value-expr; using CAST allows a specific data type to be specified. However, this is not required and is rarely necessary.

You can define an AUTOMATIC INSERT column to automatically receive data during an insert operation. The data is stored like any other column, but the column is read-only. Because AUTOMATIC columns are treated as read-only columns, they cannot appear in the column list for an insert operation nor be modified by an update operation. AUTOMATIC UPDATE columns can have an associated default value that will be used when the row is inserted.

Suppose that you want to store the current time stamp of a transaction and supply a unique numeric value for an order number. In addition, when the row is updated (the order is altered), you want a new time stamp to be written to the LAST_UPDATED column. You could write an application to supply this information, but you could not guarantee the desired behavior. For instance, a user with access to the table might update the table with interactive SQL and forget to enter a new time stamp to the LAST_UPDATED column. If you use an AUTOMATIC column instead, it can be defined so that columns automatically receive data during an insert operation. The data is sorted like any other column, but the column is read-only. See Example 2 in the Examples section.

DEFAULT value-expr

Provides a default value for a column if a row that is inserted does not include a value for that column.

If you do not specify a default value, a column inherits the default value from the domain. If you do not specify a default value for either the column or the domain, SQL assigns NULL as the default value.

You cannot specify a default value if you specify a computed column.

Remember that the default value for a column is not the same as the missing value that you can specify using the RDO interface. See the *Oracle Rdb7 Guide to Database Design and Definition* for a discussion of the difference between a default value and a missing value.

constraint-attributes

Although the constraint attribute syntax, shown in Table 3–4, provides 11 permutations as required by the SQL92 or SQL99 standard, they equate to the following three options:

- INITIALLY IMMEDIATE NOT DEFERRABLE

  Specifies that evaluation of the constraint must take place when the INSERT, DELETE, or UPDATE statement executes. If you are using the SQL99, SQL92, MIA, or ORACLE LEVEL1 dialect, this is the default.

  This clause is the same as the NOT DEFERRABLE option provided in previous releases of Oracle Rdb.

- INITIALLY DEFERRED DEFERRABLE

  Specifies that evaluation of the constraint be deferred (using the SET CONSTRAINT ALL statement or the SET TRANSACTION statement with the EVALUATING clause), but by default it is evaluated after the INSERT, DELETE, or UPDATE statement executes. This option is new for Oracle Rdb release 7.1.

- INITIALLY IMMEDIATE DEFERRABLE

  Specifies that evaluation of the constraint can take place at any later time. Unless otherwise specified, evaluation of the constraint takes place as the COMMIT statement executes. You can use the SET ALL CONSTRAINTS statement to have all constraints evaluated earlier. See the description of the SET ALL CONSTRAINTS statement in the *Oracle Rdb7 SQL Reference Manual* for more information. This clause is the same as the DEFERRABLE option provided in previous releases of Oracle Rdb. When using this dialect, Oracle Rdb displays a deprecated feature message for all constraints defined without specification of one of the constraint attributes.

  If you are using the default SQLV40 dialect, this is the default constraint attribute.

enable-disable-clause
Allows you to enable or disable all constraints, specified constraints, a primary key, or a unique column name, as described in the following list. By default, table and column constraints added during a create table operation are enabled.

- DISABLE ALL CONSTRAINTS

  All table and column constraints for this table are disabled. (No error is raised if no constraints are defined on the table.)

- ENABLE ALL CONSTRAINTS

  All table and column constraints for this table are enabled. (No error is raised if no constraints are defined on the table.)

- DISABLE CONSTRAINT constraint-name

  The named constraint is disabled. The named constraint must be a table or column constraint for the table.

- ENABLE CONSTRAINT constraint-name

  The named constraint is enabled. The named constraint must be a table or column constraint for the table.

- DISABLE PRIMARY KEY

  The primary key for the table is disabled.

- ENABLE PRIMARY KEY

  The primary key for the table is enabled.

- DISABLE UNIQUE (column-name)

The matching UNIQUE constraint is disabled. The columns listed must be columns in the table.

- ENABLE UNIQUE (column-name)

  The matching UNIQUE constraint is enabled. The columns listed must be columns in the table.

LOGGING
NOLOGGING

The LOGGING clause specifies that the CREATE TABLE statement should be logged in the recovery-unit journal file (.ruj) and after-image journal file (.aij).

The NOLOGGING clause specifies that the CREATE TABLE statement should not be logged in the recovery-unit journal file (.ruj) and after-image journal file (.aij).

The LOGGING clause is the default.

See Section 2.2.8 for information on the advantages and disadvantages of specifying these clauses and for the implications that using the NOLOGGING keyword has on database recovery.

Usage Notes

- When a constraint is disabled, it is not evaluated by the INSERT, UPDATE, or DELETE statement.
- The RMU Verify command with the Constraint qualifier ignores any disabled constraint unless that constraint is named with the Constraint qualifier and the Constraint option. If the Constraint qualifier specifies all constraints (no options are specified), or if it specifies a specified table, all disabled constraints are ignored. This allows you to check a disabled constraint periodically without the need to reenable it, which might be useful if the overhead of checking the constraint during operating hours is too expensive, or if it is already being enforced by the application.
- The following usage notes apply to AUTOMATIC columns:
  - When the column is omitted from an insert operation, a column default and an automatic column provide similar functions. However, there are distinctions, as follows:
    - AUTOMATIC columns can execute complex expressions, including select expressions and function calls.
    - AUTOMATIC columns cannot be referenced during an insert operation, because they are read-only to applications.

    * AUTOMATIC columns can be written during an update operation.
    * When you use an AUTOMATIC column, you do not provide the data type for the column.
- Note the following differences between using COMPUTED BY columns and AUTOMATIC columns:
    * COMPUTED BY columns use no space in the row, AUTOMATIC columns do.
    * A COMPUTED BY column is evaluated when the row is fetched, such as when a SELECT, UPDATE, or DELETE statement references the column name. An AUTOMATIC column is evaluated during an INSERT or UPDATE statement. A calculated value is written to a column in the row, and the value returned by a SELECT statement is the stored column value.

      For example, a column defined as COMPUTED BY CURRENT_DATE returns the date when the query is executed. A selected column that is AUTOMATIC INSERT AS CURRENT_DATE returns the date when the INSERT was performed, which might be different from the date when the query is executed.
- Note the following differences between using an AUTOMATIC column and a trigger on the table:
    * In an insert operation, an AFTER INSERT TRIGGER trigger can provide AUTOMATIC column functionality. However, AUTOMATIC columns can help eliminate the overhead of a trigger and so simplify table management.
    * Trigger actions cannot modify a row being updated, because this leads to a recursive trigger action. AUTOMATIC UPDATE columns are evaluated prior to the trigger and constraint execution.
- If the data written to the table with an AUTOMATIC column is incorrect, you can temporarily suspend the read-only attribute of the column by issuing the SET FLAGS 'AUTO_OVERRIDE' statement if you have the DBADMIN privilege on the database. Then, you can execute an update query to correct the incorrect data. See the SET FLAGS Statement for more information and an example.
- You can create an index on an AUTOMATIC column. AUTOMATIC columns are identical to other columns, except that Oracle Rdb, not a user application, assigns the value.

- The following usage notes apply to UNIQUE constraints:
  - Oracle Rdb provides an SQL:1999-compliant UNIQUE constraint. This type of constraint excludes NULL columns from the UNIQUE comparison. This effectively allows sets of columns to be UNIQUE or NULL.

    This type of constraint is created by default when the SQL dialect is set to SQL89, MIA, ORACLE LEVEL1, SQL99, or SQL92. The default dialect is SQLV40. Oracle Corporation recommends that you set the dialect to SQL99 (or one of the listed dialects) before using the CREATE TABLE statement (or ALTER TABLE statement) to add UNIQUE constraints to tables.

    Note: The UNIQUE semantics are used at run time under any selected dialect. That is, the table must be created under the listed dialects to have the new style of UNIQUE constraints enabled.

  - The new UNIQUE constraint implementation, in addition to conforming to the Database Language SQL standard, also provides improved performance for single row insert operations. This is made possible by eliminating checks for NULL values from the selection expression and thus simplifying the optimization for unique checking.

    Here is a comparison of the old and new optimizer strategies. In this example, a UNIQUE constraint ("UNIQUE_A") and index on column A are used to check for uniqueness during an INSERT statement. Note that the optimizer chooses a full range search of the index (for example, [0:0]):

```
~S: Constraint "UNIQUE_A" evaluated
Cross block of 2 entries
  Cross block entry 1
    Conjunct        Firstn  Get     Retrieval by DBK of relation T_UNIQUE
  Cross block entry 2
    Conjunct        Aggregate-F2    Conjunct
    Index only retrieval of relation T_UNIQUE
      Index name  T_UNIQUE_INDEX_A [0:0]
```

With the simplified UNIQUE constraint ("UNIQUE_B"), the optimizer can use a direct lookup of the index (that is, [1:1]), which reduces the I/O to the index to perform the constraint evaluation:

```
~S: Constraint "UNIQUE_B" evaluated
Cross block of 2 entries
  Cross block entry 1
    Conjunct        Firstn  Get     Retrieval by DBK of relation T_UNIQUE
  Cross block entry 2
```

```
Conjunct        Aggregate-F2    Index only retrieval of relation T_UNIQUE
  Index name  T_UNIQUE_INDEX_B [1:1]
```

– In prior versions, the UNIQUE constraint restricted columns to a single NULL value. To retain this behavior, use the SET DIALECT 'SQLV40' statement before creating new tables or altering existing tables to add UNIQUE constraints.

– UNIQUE constraints created in previous versions of Oracle Rdb will perform as in previous versions. Interfaces such as RDO or the Oracle CDD / Repository will continue to define the older style UNIQUE constraint. Database EXPORT and IMPORT will retain the UNIQUE constraint characteristics as defined by the database administrator, regardless of the defined dialect setting.

Note: The RMU Extract command with the Item=Table qualifier does not distinguish between the old and new UNIQUE constraints in this release of Oracle Rdb. You must modify the generated SQL script to establish the appropriate dialect before using the script to create a database.

– Because this new style of UNIQUE constraints is a relaxation of the UNIQUE rules, it is possible to drop the old style UNIQUE constraint and redefine the constraint under the SQL92 or similar dialect.

  Note that this meaning of UNIQUE (that is, excluding NULL from the uniqueness test) does not apply to the UNIQUE index. The UNIQUE index still does not allow duplicate entries for NULL. If a UNIQUE index is currently defined that assists the UNIQUE constraint optimization, then the database administrator may want to drop the index and make it a non-UNIQUE index so that multiple NULLs can be stored. The UNIQUE constraint still enforces the uniqueness of the data.

– You can use the SQL SHOW TABLE command to determine which type of UNIQUE constraint is in use. See Example 3 in the Examples section.

– As a side effect of this change to UNIQUE constraints, Oracle Rdb also recognizes a larger class of CHECK constraints as being uniqueness checks. The main benefit is that these constraints are no longer executed when a DELETE statement is executed for the table, because DELETE statements do not affect the uniqueness of the remaining rows. For example:

```
SQL> CREATE TABLE T_USER_UNIQUE_NEW
```

```
cont>     A INTEGER,
cont>     B INTEGER,
cont>     CONSTRAINT UNIQUE_AB_NEW
cont>         CHECK ((SELECT COUNT(*)
cont>                 FROM T_USER_UNIQUE_NEW T2
cont>              WHERE T2.A = T_USER_UNIQUE_NEW.A andcont>
T2.B = T_USER_UNIQUE_NEW.B) <= 1)
cont>             NOT DEFERRABLE
cont>     );
```

In previous versions of Oracle Rdb, only equality with 1 was recognized as a uniqueness constraint. In this example a comparison of LESS THAN or EQUAL TO 1 also qualifies as a uniqueness constraint.

Examples

Example 1 Enabling and Disabling Constraints While Creating a Table

```
SQL> SET DIALECT 'SQL92';
SQL> CREATE TABLE TT
cont> (A INTEGER CONSTRAINT A1 UNIQUE,
cont> CONSTRAINT A2 UNIQUE (A),
cont> CONSTRAINT A3 PRIMARY KEY (A))
cont> ENABLE CONSTRAINT A1
cont> DISABLE CONSTRAINT A2;
```

Example 2 Using AUTOMATIC Columns

```
SQL> CREATE TABLE ORDER_HEADER
cont> (ORDER_NUMBER   AUTOMATIC INSERT AS NEW_ORDER.NEXTVAL,
cont>  ORDER_DATE     AUTOMATIC INSERT AS CURRENT_TIMESTAMP,
cont>  LAST_UPDATED   AUTOMATIC UPDATE AS CURRENT_TIMESTAMP
cont>                 DEFAULT NULL,
cont>  CUSTOMER_NUMBER INTEGER,
cont>  ORDER_TOTAL     MONEY CHECK (ORDER_TOTAL > 0.0));
```

Example 3 SHOW TABLE Output for Old and New UNIQUE Constraints

```
SQL> -- This first example is a UNIQUE constraint created when
SQL> -- the default dialect is used (SQLV40). A new description
SQL> -- follows the "Unique constraint" text, explaining the
SQL> -- interpretation of null values.
SQL> SHOW TABLE (CONSTRAINT) T_UNIQUE
Information for table T_UNIQUE
```

```
Table constraints for T_UNIQUE:
T_UNIQUE_UNIQUE_B_A
 Unique constraint
         Null values are considered the same
Table constraint for T_UNIQUE
Evaluated on UPDATE, NOT DEFERRABLE
Source:
             UNIQUE (b,a)
.
.
.
SQL> -- This second example is a UNIQUE constraint created
SQL> -- when the dialect was set to 'SQL92', and the description
SQL> -- here indicates that all null values are considered
SQL> --  distinct.
SQL> SHOW TABLE (CONSTRAINT) T_UNIQUE2;
Information for table T_UNIQUE2
Table constraints for T_UNIQUE2:
T_UNIQUE2_UNIQUE_B_A
Unique constraint
                Null values are considered distinct
Table constraint for T_UNIQUE2
 Evaluated on UPDATE, NOT DEFERRABLE
 Source:        UNIQUE (b,a)
.
.
.
```

CREATE TRIGGER Statement

Creates triggers for a specified table. A **trigger** defines the actions to occur before or after the table is updated (by a write operation such as an INSERT, DELETE, or UPDATE statement). The trigger is associated with a single table, takes effect at a specific time for a particular type of update, and causes one or more triggered actions to be performed. If the trigger specifies multiple actions, each action is performed in the order in which it appears within the trigger definition.

With triggers, you can define useful actions such as:

- Cascading deletes

  Deleting a row from one table causes additional rows to be deleted from other tables that are related to the first table by key values.

- Cascading updates

  Updating a row in one table causes additional rows to be updated in other tables that are related to the first table by key values. These updates are commonly limited to the key fields themselves.

- Summation updates

  Updating a row from one table causes a value in a row of another table to be updated by being increased or decreased.

- Hidden deletes

  Moving rows to a parallel table that is not otherwise used by the database causes rows to be deleted from the original table.

Note: Combinations of table-specific constraints and appropriately defined triggers, by themselves, are not sufficient to guarantee that database integrity is preserved when the database is updated. If integrity is to be preserved, table-specific constraints and triggers must be used in conjunction with a common set of update procedures that ensure completely reproducible and consistent retrieval and update strategies.

The CREATE TRIGGER statement adds the trigger definition to the physical database.

If you did not attach to the database by a path name, the trigger definition is not stored in the repository. This causes an inconsistency between the definitions in the database and the repository. Therefore, you must define the triggers again whenever you restore the database metadata from the repository using the INTEGRATE statement.

A **triggered action** consists of an optional predicate and some triggered statements. If specified, the predicate must evaluate to true for the triggered statements in the action to execute. Each triggered statement is executed in the order in which it appears within the triggered action clause.

The triggered statement can be:

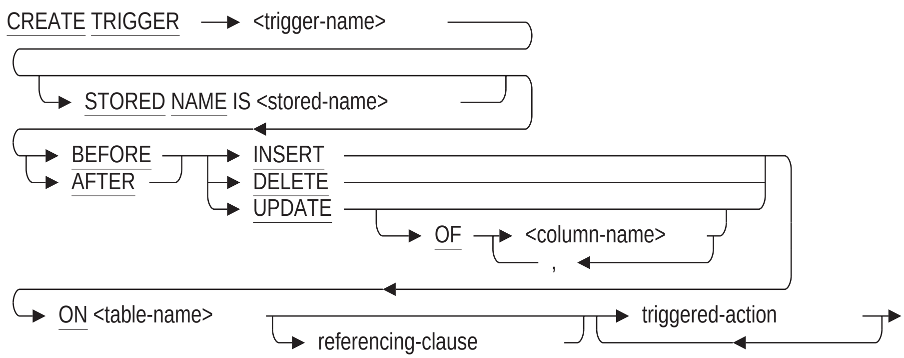
- A DELETE statement
- An UPDATE statement
- An INSERT statement
- A CALL statement
- A SIGNAL statement
- A TRACE statement
- An ERROR statement

Environment

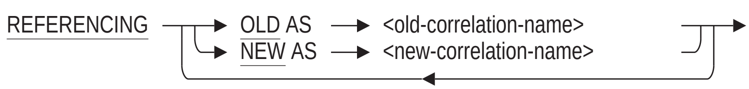
You can use the CREATE TRIGGER statement:

- In interactive SQL
- Embedded in host language programs to be precompiled
- As part of a procedure in an SQL module
- In dynamic SQL as a statement to be dynamically executed

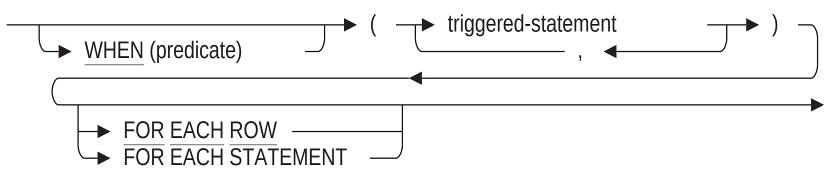
Format



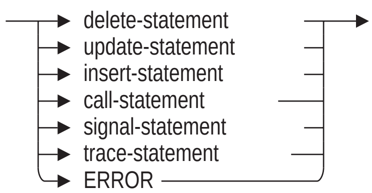
referencing-clause =



triggered-action =



triggered-statement =



Arguments

FOR EACH ROW
FOR EACH STATEMENT
Specifies whether the triggered action is evaluated once per triggering statement, or for each row of the subject table that is affected by the triggering statement.

If you specify FOR EACH STATEMENT, then the triggered action is evaluated only once, and row values are not available to the triggered action.

The FOR EACH STATEMENT clause is the default.

call-statement
Specifies the stored procedure to invoke. You can only call procedures with IN parameters. Operations on the triggering table are not permitted due to possible side effects and recursive calls.

signal-statement
Specifies that the signaled SQLSTATE status parameter is to be passed back to the application or SQL interface and that the current routine and all calling routines are to be terminated. This provides a more complete error mechanism than is provided by the ERROR clause.

trace-statement
Allows applications to add triggers to log information when trace logging is active.

Usage Notes

- See the usage notes for the CREATE TABLE Statement for information on how UNIQUE constraint behavior has changed.

- Oracle Rdb tracks language semantics for each trigger. If the language semantics are altered, the trigger is invalidated and must be re-created. The following semantics are fixed at data definition time:
  - SELECT * FROM table-name

    The asterisk (*) expands to a column list
  - INSERT INTO table-name VALUES (...)

    The column list defaults to the current names and order of the tables
  - Natural join

    The matching names are used for equijoins

  For example:

```
SQL> CREATE TRIGGER AFTER_T AFTER INSERT ON T
cont> (INSERT INTO S VALUES (T.ID, T.SEQ)) FOR EACH ROW;
SQL> ALTER TABLE S ADD COLUMN P REAL;
%RDB-W-META_WARN, metadata successfully updated with the reported warning
- RDMS-W-TRIG_LANGSEMEXI, table used by trigger with language dependency -
trigger invalid on COMMIT
SQL> COMMIT;
.
.
.
SQL> INSERT INTO T VALUES (0,0);
%RDB-E-TRIG_REQ_ERROR, error encountered by a request using triggers
- RDMS-E-TRG_INVALID, trigger can not be invoked - it is marked invalid
-RDMS-E-TRIG_ERROR, trigger AFTER_T forced an error
```

Examples

Example 1 Creating a Trigger

```
SQL> -- Create new table to record changes made to
SQL> -- EMPLOYEES table
SQL> CREATE TABLE AUDIT_TRAIL
cont> (LOG DATE VMS,
cont>  PERSON CHAR(31),
cont> TBL_NAME CHAR(10),
cont> OPER CHAR(1));
SQL> COMMIT;

SQL> -- Create a trigger so that each time
SQL> -- an INSERT operation is performed,
```

```
SQL> -- a record is stored in the AUDIT_TRAIL table.
SQL> CREATE TRIGGER EMPS_TRIGGER
cont> AFTER INSERT
cont> ON EMPLOYEES
cont> (INSERT INTO AUDIT_TRAIL
cont> VALUES (CURRENT_TIMESTAMP,
cont>         CURRENT_USER, 'EMPLOYEES', 'I'))
cont> FOR EACH STATEMENT;
SQL> -- The AUDIT_TRAIL table currently has no records.
SQL> SELECT * FROM AUDIT_TRAIL;
0 rows selected
SQL> -- Insert a record into EMPLOYEES
SQL> INSERT INTO EMPLOYEES
cont> (EMPLOYEE_ID, LAST_NAME)
cont> VALUES ('00964', 'FRENCH');
1 row inserted
SQL> -- See if trigger updated the AUDIT_TRAIL table.
SQL> SELECT * FROM AUDIT_TRAIL;
 LOG                     PERSON                           TBL_NAME     OPER
 17-JUN-1998 15:04:31.43   STEWART                          EMPLOYEES    I
1 row selected
```

CREATE USER Statement

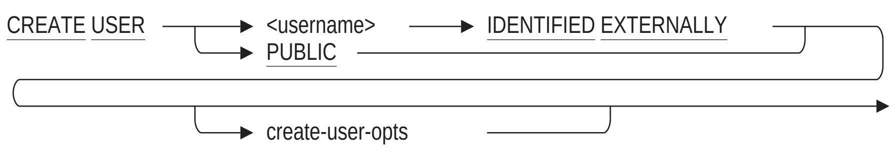
Creates a special security profile entry to identify a database user. That user can be granted roles, which in turn provide access to database objects.

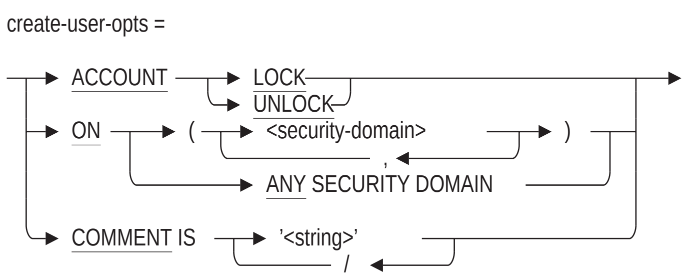
Environment

You can use the CREATE USER statement:

- In interactive SQL
- Embedded in host language programs to be precompiled
- As part of a procedure in an SQL module
- In dynamic SQL as a statement to be dynamically executed

Format





Arguments

username
An existing user (that matches an existing user name and UIC). To use lowercase characters or characters not in the SQL repertoire, enclose the user name in double quotation marks (").

IDENTIFIED EXTERNALLY
Indicates that the user will be authenticated through the operating system, not using a password.

PUBLIC
Explicitly creates a PUBLIC security profile entry in the database. The special user PUBLIC exists implicitly, but by explicitly creating a PUBLIC entry, you can associate a role and security profile with the PUBLIC user. This gives you control over anonymous users who access the database.

ACCOUNT LOCK
ACCOUNT UNLOCK
The ACCOUNT LOCK clause disables access to the database by the user for whom the CREATE USER statement is being applied. The ACCOUNT UNLOCK clause allows that user access to the database.

The ACCOUNT UNLOCK clause is the default.

ON (security-domain)
ON ANY SECURITY DOMAIN
The ON security-domain clause specifies one or more security domains from which the user can access the database. If you do not specify an ON clause, then the specified user can access the database from the current security domain only (as determined by the database server). The ON ANY SECURITY DOMAIN clause specifies that this user can access the database from any domain.

The ON clause can also be used with the PUBLIC user. For instance, if SMITH is not registered as a user in the database, then he will implicitly inherit the attributes assigned to PUBLIC. Anonymous users can be restricted to security domains using the ON clause.

COMMENT IS ’string’
Adds a comment about the user. SQL displays the text of the comment when it executes a SHOW USERS statement. Enclose the comment in single quotation marks (’) and separate multiple lines in a comment with a slash mark (/).

Usage Notes

- You must have the SECURITY privilege on the database to create a user.
- The special user PUBLIC exists implicitly. However, the CREATE USER statement can be used to create an explicit PUBLIC entry so that roles can be associated with the PUBLIC user. This allows control of anonymous users who access the database.
- You can display existing users defined for a database by issuing a SHOW SYSTEM USERS or SHOW USERS statement.
- Oracle Rdb rightcases all user and role names that are identified externally. **Rightcasing** means that the names are stored as they are specified in the operating system registry or authorization database. On OpenVMS, all names are converted to uppercase, even if you use delimiters.

Examples

Example 1 Creating a New User and Locking Her Account

```
SQL> CREATE USER munroy IDENTIFIED EXTERNALLY
cont> ACCOUNT LOCK
cont> COMMENT IS 'User munroy starts job on'/
cont> 'May 1, 1998.  Unlock when she starts';
SQL> -- If user munroy attempts to attach to the
SQL> -- database, an error is returned:
SQL> ATTACH 'FILENAME mf_personnel.rdb';
%SQL-F-ERRATTDEC, Error attaching to database mf_personnel.rdb
-RDB-E-NO_PRIV, privilege denied by database facility
```

DECLARE TRANSACTION Statement

Specifies the characteristics for a default transaction. A **transaction** is a group of statements whose changes can be made permanent or undone only as a unit.

A transaction ends with a COMMIT or ROLLBACK statement. If you end the transaction with the COMMIT statement, all changes made to the database by the statements are made permanent. If you end the transaction with the ROLLBACK statement, the statements do not take effect.

The characteristics specified in a DECLARE TRANSACTION statement affect all transactions (except those started by the SET TRANSACTION statement) until you issue another DECLARE TRANSACTION statement. The characteristics specified in a SET TRANSACTION statement affect only that transaction.

A DECLARE TRANSACTION statement does not start a transaction. The declarations made in a DECLARE TRANSACTION statement do not take effect until SQL starts a new transaction. SQL starts a new transaction with the first executable data manipulation or data definition statement following a DECLARE TRANSACTION, COMMIT, or ROLLBACK statement. Following a COMMIT or ROLLBACK statement, SQL applies the transaction characteristics you declared for the transaction that just ended to the next one you start.

In addition to the DECLARE TRANSACTION statement, you can specify the characteristics of a transaction in one of two ways:

- If you specify the SET TRANSACTION statement, the declarations in the statement take effect immediately and SQL starts a new transaction.
- You can retrieve and update data without declaring or setting a transaction explicitly. If you omit both the DECLARE TRANSACTION and SET TRANSACTION statements, SQL automatically starts a transaction (using the read / write option) with the first executable data manipulation or data definition statement following a COMMIT or ROLLBACK statement.

See the Usage Notes for the DECLARE TRANSACTION statement in the *Oracle Rdb7 SQL Reference Manual* for examples of when you would want to use the DECLARE TRANSACTION statement instead of the SET TRANSACTION statement.

You can specify many options with the DECLARE TRANSACTION statement, including:

- A transaction mode (READ ONLY, READ WRITE, or BATCH UPDATE)

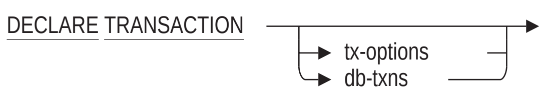
- A lock specification clause (RESERVING options)
- A wait mode (WAIT or NOWAIT)
- An isolation level
- A constraint evaluation specification clause
- Multiple sets of all the preceding options for each database involved in the transaction (ON, AND ON)

Environment

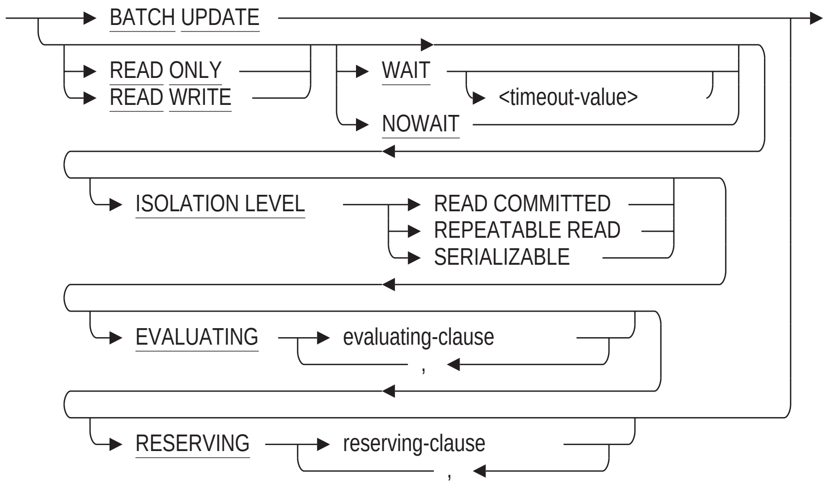
You can use the DECLARE TRANSACTION statement:

- In interactive SQL
- Embedded in host language programs to be precompiled
- In a context file
- As part of the DECLARE section in an SQL module
- In dynamic SQL as a statement to be dynamically executed

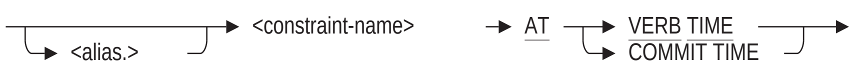
Format



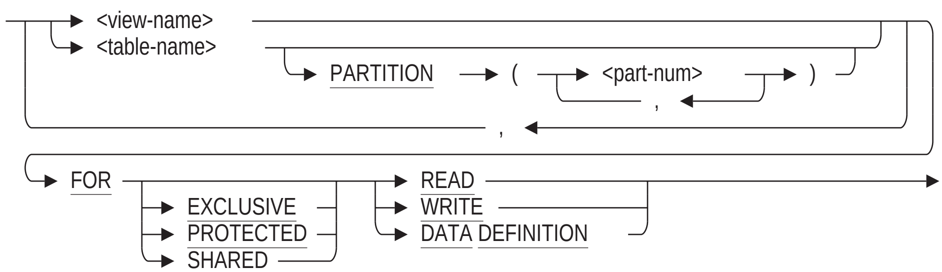
tx-options =



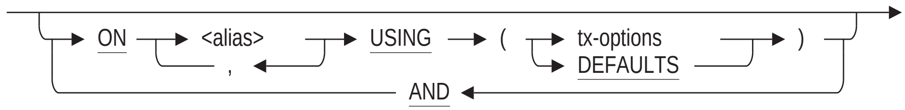
evaluating-clause =



reserving-clause =



db-txns =



Arguments

PARTITION (part-num)
Specifies the partition number for the partition to be reserved or locked. Only values in the RDB$STORAGE_MAP_AREAS table in the RDB$ORDINAL_POSITION column can be specified. SQL ignores duplicate part-num values in the reserving clause. Partitions that you do not list are reserved for shared access by default.

The PARTITION clause and the DATA DEFINITION clause are mutually exclusive; you cannot specify one if you specify the other.

Usage Notes

- By default, a transaction that reserves a table for EXCLUSIVE access does not reserve the LIST (segmented string) area for exclusive access. Because the LIST area is usually shared by many tables, SHARED access is assumed by default to permit updates to the other tables.

  This means that when you perform an import operation, or an application updates a table reserved for EXCLUSIVE access, you might notice that the snapshot storage area (.snp) grows. This is because of the I/O to the LIST area that is performed by default when SHARED WRITE mode is in use.

  However, if you attach to the database using an SQL ATTACH or IMPORT statement and you specify the RESTRICTED ACCESS clause, then all storage areas are accessed in EXCLUSIVE mode. Use this clause to eliminate the snapshot I/O and related overhead if you are performing a lot of I/O to the LIST storage areas (for example, when you are restructuring the database or dropping a large table containing LIST OF BYTE VARYING columns and data).

Examples

Example 1 Reserving a Partition

```
SQL> -- Determine the ordinal position of the EMPLOYEES
SQL> -- partitions.
SQL> SELECT RDB$MAP_NAME, RDB$AREA_NAME, RDB$ORDINAL_POSITION
cont> FROM RDB$STORAGE_MAP_AREAS
cont> WHERE RDB$MAP_NAME='EMPLOYEES_MAP';
 RDB$MAP_NAME                     RDB$AREA_NAME
   RDB$ORDINAL_POSITION
 EMPLOYEES_MAP                    EMPIDS_LOW
                     1
 EMPLOYEES_MAP                    EMPIDS_MID
                     2
 EMPLOYEES_MAP                    EMPIDS_OVER
                     3
3 rows selected
SQL> --
SQL> -- Reserve EMPIDS_MID and EMPIDS_OVER for
SQL> -- exclusive write.
SQL> --
SQL> DECLARE TRANSACTION
cont> RESERVING EMPLOYEES PARTITION (2,3)
cont> FOR EXCLUSIVE WRITE;
```

DELETE Statement

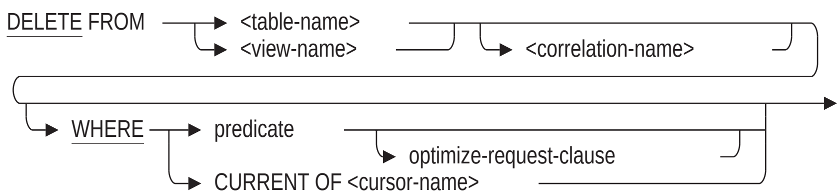
Deletes a row from a table or view.

Environment

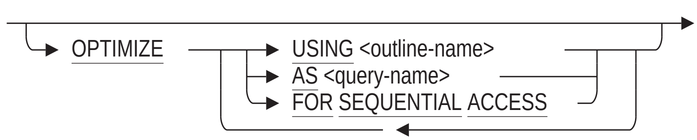
You can use the DELETE statement:

- In interactive SQL
- Embedded in host language programs to be precompiled
- As part of a procedure in an SQL module
- In dynamic SQL as a statement to be dynamically executed

Format



optimize-request-clause =



Arguments

DELETE FROM

Beginning with this release of Oracle Rdb, the FROM keyword is optional in the DELETE statement.

Examples

Example 1 Using the DELETE Statement Without the FROM Keyword

```
SQL> DELETE EMPLOYEES E
cont> WHERE EXISTS (SELECT *
cont>              FROM   SALARY_HISTORY S
cont>              WHERE  S.EMPLOYEE_ID=E.EMPLOYEE_ID
cont>              AND    S.SALARY_AMOUNT > 75000
cont>             );
```

DROP PROFILE Statement

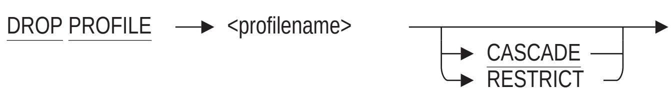
Drops a profile definition.

Environment

You can use the DROP PROFILE statement:

- In interactive SQL
- Embedded in host language programs
- As part of a procedure in an SQL module
- In dynamic SQL as a statement to be dynamically executed

Format



Arguments

CASCADE
This option causes all user definitions to be altered to remove the reference to this profile.

RESTRICT
If the profile is used by a user in the database, the DROP PROFILE statement will fail. This is the default.

Usage Notes

- Profile names are, by default, in uppercase. If they were defined in mixed case or with other special characters, use the SET DIALECT ’SQL92’ or ’ORACLE LEVEL1’ or ’SQL99’ or SET QUOTING RULES ’SQL92’ or ’ORACLE LEVEL1’ or ’SQL99’ statement to enable delimited identifiers. Then, use quotation marks (" ") around the name in the DROP PROFILE statement.

Examples

Example 1 Using Delimited Identification Mixed-Case Profile Names

```
SQL> DROP PROFILE Decision_Support;
%RDB-E-NO_META_UPDATE, metadata update failed
-RDMS-E-PRFNEXISTS, a quota does not exist with the name "DECISION_SUPPORT"
SQL> SET DIALECT 'SQL92';
SQL> DROP PROFILE "Decision_Support";
SQL> COMMIT;
```

DROP ROLE Statement

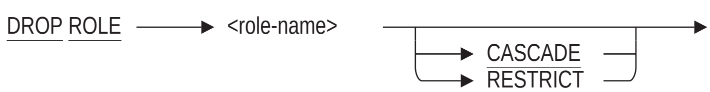
Drops a role previously created with the CREATE ROLE statement.

Environment

You can use the DROP ROLE statement:

- In interactive SQL
- Embedded in host language programs to be precompiled
- As part of a procedure in an SQL module
- In dynamic SQL as a statement to be dynamically executed

Format



Arguments

role-name
An existing role-name in the database (such as one created with the CREATE ROLE statement). You cannot specify one of the predefined roles. See the Usage Notes for details.

CASCADE
Drops the specified role from the database and deletes all references to this role that exist in other roles and access control lists (ACLs).

RESTRICT
Drops the specified role. If there are any references to this role in another role or ACL, then the DROP ROLE statement fails.

The RESTRICT clause is the default.

Usage Notes

- You must have the SECURITY privilege on the database to drop a role.
- The special roles BATCH, DIALUP, INTERACTIVE, LOCAL, NETWORK, and REMOTE are granted by the OpenVMS operating system when the user process is created. Therefore, these roles are reserved names and cannot be used as the role-name in the DROP ROLE statement.

Examples

Example 1 Dropping a Role from the Database

```
SQL> SHOW ROLES;
Roles in database with filename mf_personnel.rdb
     DOCUMENTATION
SQL> DROP ROLE DOCUMENTATION RESTRICT;
SQL> SHOW ROLES;
Roles in database with filename mf_personnel.rdb
 No Roles Found
```

DROP SEQUENCE Statement

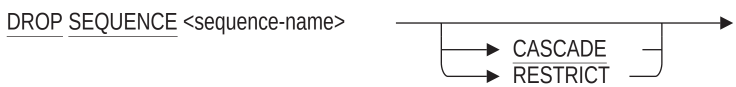
Drops a specified sequence.

Environment

You can use the DROP SEQUENCE statement:

- In interactive SQL
- Embedded in host language programs to be precompiled
- As part of a procedure in an SQL module
- In dynamic SQL as a statement to be dynamically executed

Format



Arguments

sequence-name
An existing sequence name in the database. To specify lowercase characters or characters not in the SQL repertoire, enclose the user name in single quotation marks (’).

CASCADE
RESTRICT
The CASCADE clause specifies that you want SQL to invalidate all objects that refer to the sequence and then delete the sequence definition. If you delete a sequence referenced by a stored routine or trigger with a routine or language-semantic dependency, SQL also marks the affected stored routine or trigger as invalid.

The RESTRICT clause prevents the removal of a sequence definition (the DROP SEQUENCE statement fails) when the sequence is referenced by any other object within the Oracle Rdb database.

The RESTRICT clause is the default.

Usage Notes

- You must have the DROP database privilege on the sequence to drop a sequence from a database.
- When you drop a sequence, the reserved space in the database root file becomes available for reuse by the next sequence created.

Examples

Example 1 Dropping a Sequence

```
SQL> SHOW SEQUENCE;
Sequences in database with filename mf_personnel.rdb
     EMPID
SQL> DROP SEQUENCE EMPID;
SQL> SHOW SEQUENCE;
Sequences in database with filename mf_personnel.rdb
 No Sequences Found
SQL>
```

DROP SYNONYM Statement

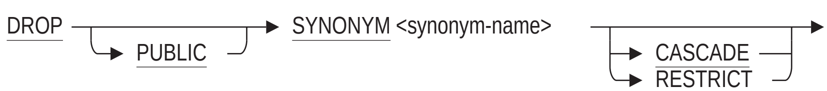
Drops a synonym definition.

Environment

You can use the DROP SYNONYM statement:

- In interactive SQL
- Embedded in host language programs
- As part of a procedure in an SQL module
- In dynamic SQL as a statement to be dynamically executed

Format



Arguments

synonym-name
The name of an existing synonym you want to drop.

PUBLIC
This optional clause is provided for compatibility with the Oracle database server. It is currently not used by Oracle Rdb. Its presence or absence may be used by future releases. Oracle Corporation recommends you use the PUBLIC keyword in applications.

CASCADE
Specifies that you want SQL to delete the synonym definition even if other database objects reference this name. This might later cause errors when executing queries. Stored functions, stored procedures, and triggers that reference this name will be marked as invalid.

RESTRICT

Specifies that you want SQL to abort the DROP statement if it detects any database object referencing this name. This is the default.

Usage Notes

- An error is generated if this statement is used on a database that has not been enabled for synonyms. See the ALTER DATABASE ... SYNONYMS ARE ENABLED clause.
- You must have database DROP privilege to execute the DROP SYNONYM statement.
- You must have REFERENCES privilege on the referenced object to drop a synonym for that object. Because domains do not have access control, no other privileges are required to drop synonyms for domains.

Examples

Example 1 Dropping a Synonym

```
SQL> DROP PUBLIC SYNONYM employees CASCADE;
```

DROP USER Statement

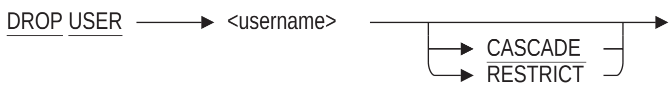
Removes the entry (such as one created with the CREATE USER statement) for a user name or special user class from the database.

Environment

You can use the DROP USER statement:

- In interactive SQL
- Embedded in host language programs to be precompiled
- As part of a procedure in an SQL module
- In dynamic SQL as a statement to be dynamically executed

Format



Argument

username
An existing user name in the database. To specify lowercase characters or characters not in the SQL repertoire, enclose the user name in single quotation marks (’).

CASCADE
RESTRICT
The CASCADE clause drops the specified user from the database and deletes all references to this user that exist in the access control lists (ACLs), modules, and schemas. If the PUBLIC user is dropped, ACLs are not processed to remove the PUBLIC entry.

The RESTRICT clause drops the specified user. If there are any references to this user in another ACL, then the DROP USER statement fails.

The RESTRICT clause is the default.

Usage Notes

- You must have the SECURITY privilege on the database to drop a user.
- You can display existing users defined for a database by issuing a SHOW SYSTEM USERS or SHOW USERS statement.

Examples

Example 1 Dropping a User

```
SQL> SHOW USER
Users in database with filename mf_personnel.rdb
     jsmith
     nstuart
SQL> DROP USER jsmith;
SQL> SHOW USER
Users in database with filename mf_personnel.rdb
     nstuart
SQL>
```

FOR (Counted) Control Statement

Executes a block of SQL statements while the FOR loop variable is incremented (or decremented) from a user-specified starting value to a user-specified ending value.

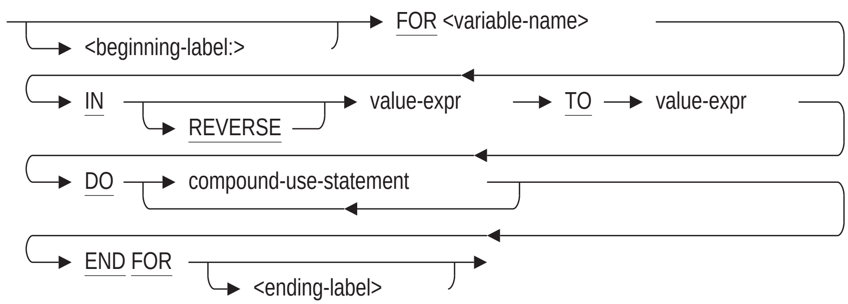
Environment

You can use the FOR counted control statement in a compound statement of a multistatement procedure:

- In interactive SQL
- Embedded in host language programs to be precompiled
- As part of a procedure in an SQL module
- In dynamic SQL as a statement to be dynamically executed

Format

counted-for-statement =



Arguments

beginning-label
Assigns a name to the FOR statement. A beginning label used with the LEAVE statement lets you perform a controlled exit from a FOR loop. A named FOR loop is called a **labeled FOR loop statement**. If you include an ending label, it must be identical to its corresponding beginning label. A beginning label must be unique within the procedure in which the label is contained.

FOR variable-name

Specifies a variable to hold a value that is incremented by 1 each time the FOR loop is executed. The variable is decremented by 1 if the REVERSE keyword is specified. The starting value for the variable is the first value expression. Execution of the FOR loop ends when the variable has been incremented (or decremented) to the value specified with the second value expression.

IN value-expression TO value-expression
IN REVERSE value-expression TO value-expression

Specifies how often the compound-use-statement should be executed. When the REVERSE keyword is not specified, the variable contained in the FOR variable-name is incremented by one at the end of each execution of the FOR loop body. When the REVERSE keyword is specified, the variable contained in the FOR variable-name is decremented by one at the end of each execution of the FOR loop body.

Both value expressions are evaluated once before the loop executes. The TO value-expression is evaluated first to ensure that references to the FOR loop variable do not cause side effects.

DO compound-use-statement

Executes a block of SQL statements once for each execution of the loop as defined by the starting and ending value expressions.

END FOR
END FOR ending-label

Marks the end of a FOR loop. If you choose to include the optional ending label, it must match exactly its corresponding beginning label. An ending label must be unique within the procedure in which the label is contained. The optional end-label argument makes the FOR loops of multistatement procedures easier to read, especially in very complex procedure blocks.

Usage Notes

- The FOR loop variable-name must exist as a declared updatable local (or global) variable.
- The FOR loop variable must be declared as a number value (TINYINT, SMALLINT, INTEGER, BIGINT, FLOAT, REAL, DOUBLE, NUMERIC, NUMBER, or DECIMAL) with no fractional portion.
- Within the body of the FOR loop, the FOR loop variable-name cannot be updated using any of the following:

- The SET statement
- The GET DIAGNOSTICS statement
- The INTO clause of the INSERT RETURNING, UPDATE RETURNING, or SELECT statements

In addition, the FOR loop variable name cannot be changed if it is the target of an INOUT or OUT parameter of the CALL statement.

In other words, the FOR loop variable behaves like a constant variable within the loop. However, outside the loop, the variable can be modified because the read-only nature of the loop variable is temporary.

- The loop body will not execute if any one of the following is true:
  - The starting value expression evaluates to NULL.
  - The ending value expression evaluates to NULL.
  - The starting value expression is greater than the ending value expression in a forward loop (one that does not contain the REVERSE keyword).
  - The starting value expression is less than the ending value expression in a reverse loop (one that contains the REVERSE keyword).

Examples

Example 1 Using a Reverse Loop

```
SQL> SET FLAGS 'TRACE';
SQL> BEGIN
cont>  DECLARE :LOOP_VAR INTEGER;
cont>  FOR :LOOP_VAR IN REVERSE 1 TO 5
cont>    DO
cont>      TRACE :LOOP_VAR;
cont>  END FOR;
cont> END;
~Xt: 5
~Xt: 4
~Xt: 3
~Xt: 2
~Xt: 1
```

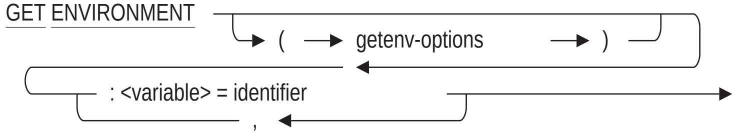
GET ENVIRONMENT Statement

Loads values defined by OpenVMS DCL symbols or logical names into locally declared SQL variables.

Environment

You can use the GET ENVIRONMENT statement in interactive SQL only.

Format



getenv-options =

Arguments

TRACE

Displays the translated string value prior to being converted to the data type of the variable. This can assist in diagnosing data conversion errors. The display will indicate if the result was derived from a local symbol, global symbol, logical name, or session value. For example:

```
SQL> GET ENVIRONMENT (TRACE)
cont>     :xx indicator :xx_ind = XX;
01: XX = XX "--" (Local)
%RDB-E-ARITH_EXCEPT, truncation of a numeric value at runtime
-COSI-F-INPCONERR, input conversion error
```

Usage Notes

- If no NULL indicator is specified and the DCL symbol is not found, an error will be reported. For example:

```
SQL> GET ENVIRONMENT
cont>     :x = THE_TIME;
%SQL-F-UNDEFVAR, Variable THE_TIME is not defined
-LIB-F-NOSUCHSYM, no such symbol
```

- If the specified symbol is not defined, and an INDICATOR is specified for the variable, the indicator will be set, but the variable will remain unchanged. For example:

```
SQL> GET ENVIRONMENT :xx indicator :xx_ind = XX;
SQL>
SQL> PRINT :xx, :xx_ind;
          XX        XX_IND
           0             1
```

  If the TRACE option is used, the value will be displayed as NULL for the symbol.

- The specified variable must be a local variable defined using the DECLARE statement. For example:

```
SQL> DECLARE :xx, :xx_ind INTEGER;
```

  The identifier is assumed to be a DCL symbol or logical name. It is first translated as a symbol name and, if that fails, it is translated as a logical name. If translation is successful, the string result is converted to the data type of the variable. The name must conform to the rules defined by the OpenVMS DCL naming conventions.

  Multiple assignments can be specified, separated by commas.

Examples

Example 1 Using the GET ENVIRONMENT Statement

```
$ emp_id = "00164"
$ SQL$
SQL> ATTACH 'FILENAME MF_PERSONNEL';
SQL> DECLARE :e CHAR(5);
SQL> GET ENVIRONMENT :e = emp_id;
SQL> SELECT last_name, first_name FROM employees WHERE employee_id = :e;
```

```
 LAST_NAME        FIRST_NAME
 Toliver          Alvin
1 row selected
SQL> ROLLBACK;
```

GRANT Statement

Creates or adds privileges to an entry to the Oracle Rdb access privilege set, called the **access control list (ACL),** for a database, table, view, column, module, external routine, or sequence. Each entry in an ACL consists of an identifier and a list of privileges assigned to the identifier:

- Each identifier specifies a user or a set of users.
- The list of privileges specifies which operations that user or user group can perform on the database, table, view, column, module, external routine, or sequence.

When a user tries to perform an operation on a database, SQL reads the associated ACL from top to bottom, comparing the identifier of the user with each entry. As soon as SQL finds the first match, it grants the rights listed in that entry and stops the search. All identifiers that do not match a previous entry "fall through" to the entry [*,*] (equivalent to the SQL keyword PUBLIC). If no entry has the identifier [*,*], then users with unmatched identifiers are denied all access to the database, table, view, column, module, external routine, or sequence.

For this reason, both the entries and their order in the list are important.

Under the Oracle Rdb default protection scheme, when you create a new database, table, view, module, external routine, or sequence, you get all access rights to that object, including DBCTRL. All other users of that object are given no access rights to it. For any tables or views created under the Oracle Rdb default protection scheme, the creator of the table or view receives all access rights to the object, including DBCTRL, and all other users receive no access rights to the object.

The DBCTRL access right enables an object's creator to grant DBCTRL to other users. See the Usage Notes for the GRANT Statement in the *Oracle Rdb7 SQL Reference Manual* for information on how you can tailor the default protection for any new tables that you create within a database.

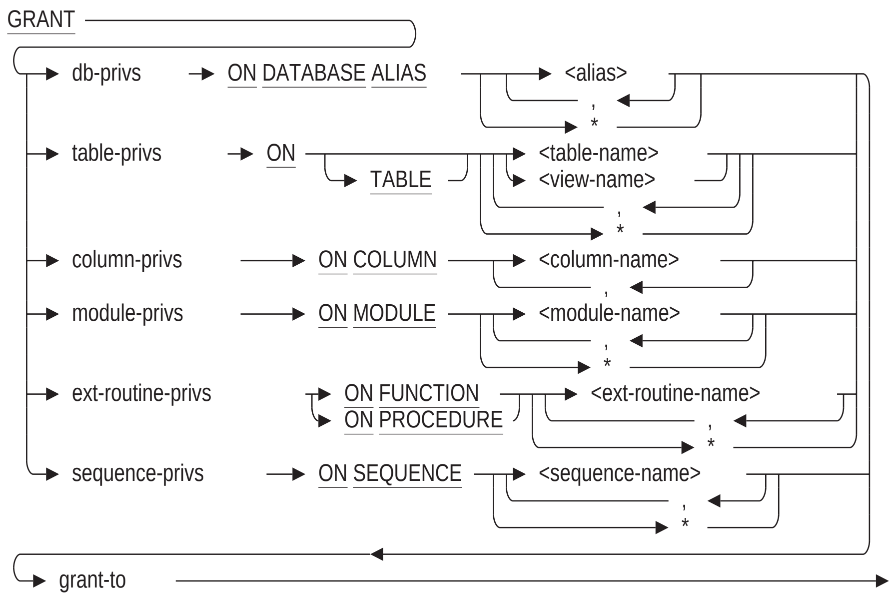
To remove privileges from or entirely delete an entry to the Oracle Rdb access privilege set for a database, table, column, module, external routine, or sequence, see the REVOKE Statement in this manual and in the *Oracle Rdb7 SQL Reference Manual*.

Environment

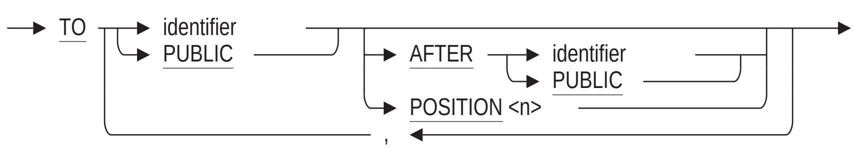
You can use the GRANT statement:

- In interactive SQL
- Embedded in host language programs to be precompiled
- As part of a nonstored procedure in a nonstored SQL module
- In dynamic SQL as a statement to be dynamically executed

Format



grant-to =



db-privs =

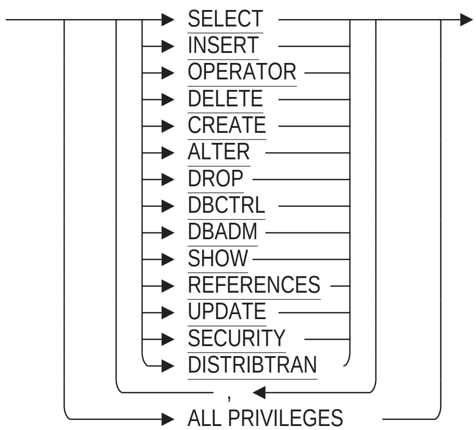
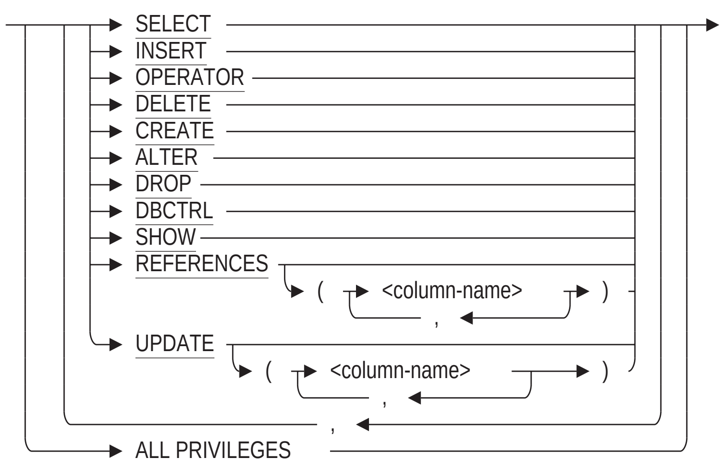
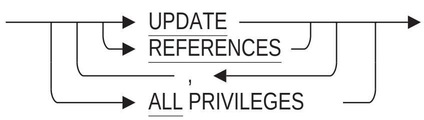


table-privs=

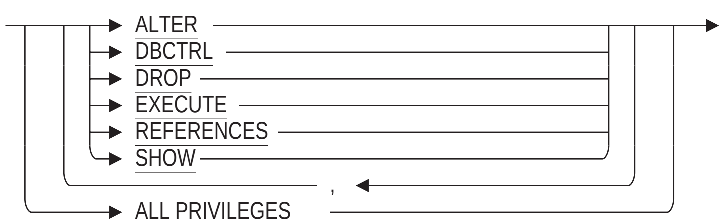


column-privs=

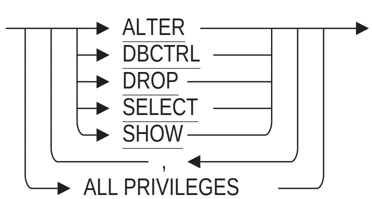


module-privs =

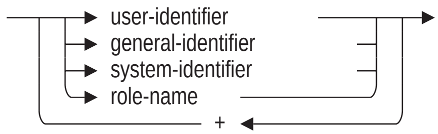
ext-routine-privs =



sequence-privs-ansi =



identifier =



Arguments

db-privs
table-privs
column-privs
module-privs
ext-routine-privs
sequence-privs

Specifies the list of privileges that you want to add to an existing ACL entry or create in a new one. The operations permitted by a given privilege keyword differ, depending on whether you granted it for a database, table, column, module, external routine, or sequence.

Table 3–5 lists the privilege keywords and their meanings for databases, tables, columns, modules, external routines, and sequences.

Table 3–5 SQL Privileges for Databases and Database Objects

Privilege	For the Access Privilege Set of a Database, Grants the Privilege to:	For the Access Privilege Set of a Table, Column, View, Sequence, Module, or External Routine, Grants the Privilege to:
ALTER	Change database parameters or change a domain.	Alter the table, index, sequence, or storage map. Alter a module or external routine. Does not apply to column privileges.
CREATE	Create a catalog, schema, table, domain, collating sequence, storage area, external routine, or module.	Create a view, trigger, index, sequence, storage map, or outline that uses a table. Does not apply to column privileges.
DBADM	Perform any data manipulation or data definition operation on any named object. Override many database privileges.	Not applicable, but syntactically allowed.

Table 3–5 (Cont.) SQL Privileges for Databases and Database Objects

Privilege	For the Access Privilege Set of a Database, Grants the Privilege to:	For the Access Privilege Set of a Table, Column, View, Sequence, Module, or External Routine, Grants the Privilege to:
DBCTRL	Create, delete, or modify an access privilege set entry for the database.	Create, delete, or modify an access privilege set entry for the table, sequence, module, or external routine. Does not apply to column privileges.
DELETE	Delete data from a table defined in the database.	Delete data from a table. Does not apply to column or sequence privileges.
DISTRIBTRAN	Run a distributed (two-phase commit protocol) transaction against the database.	Not applicable.
DROP	Delete a catalog, schema, domain, collating sequence, or path name.	Delete the table, or delete a view, index, or outline that uses a table. Delete a column, constraint, trigger, sequence, or storage map. Delete a module or external routine.
EXECUTE	Not applicable.	Allow the execution of a module or external routine. Does not apply to column, sequence, or table privileges.
INSERT	Store data in a table defined in the database.	Store data in the table. Does not apply to column or sequence privileges.
REFERENCES	Not applicable, but syntactically allowed.	Define constraints that refer to data in a table or column. Does not apply to sequence privileges.
SECURITY	Override many database privileges.	Not applicable.
SELECT	Attach to a database, read data from a table defined in the database, and set session authorization.	Read data from a table or reference the NEXTVAL and CURRVAL pseudocolumns in a sequence. Does not apply to column privileges.
SHOW	Not applicable. Syntactically allowed, but not implemented. Reserved for future versions.	Not applicable. Syntactically allowed, but not implemented. Reserved for future versions.
UPDATE	Update data in a table defined in the database.	Update data in a table or column. Does not apply to sequence privileges.

Privileges on a column are determined by the privileges defined for the table combined with those specified for the specific column ACL.

The SELECT privilege is a prerequisite for all other data manipulation privileges, except UPDATE and REFERENCES. If you do not grant the SELECT privilege, you effectively deny SELECT, INSERT, and DELETE privileges, even if they are specified in the privilege list. It is not possible for you to deny yourself the SELECT privilege.

For the SELECT, INSERT, UPDATE, and DELETE data manipulation privileges, SQL checks the ACL for the database and for the individual table before allowing access to a specific table. For example, if you are granted SELECT privilege for the EMPLOYEES table, you are not able to select rows from the table unless you also have SELECT privilege for the database that contains the EMPLOYEES table.

A user with the UPDATE privilege on the table automatically receives the UPDATE privilege on all columns in the table. To update a column, you must have the UPDATE privilege either for the column or the table. However, you can restrict the UPDATE privileges by defining them only on specific columns that you want users to be able to update, and by removing the UPDATE privilege from the table entry.

You can modify the data in a column only with the UPDATE privilege on the column and the SELECT privilege on the database.

The REFERENCES privilege lets you define a constraint for a database with ANSI/ISO-style privileges. For a database with ACL-style privileges, you need the CREATE privilege to define a constraint.

You cannot deny yourself the DBCTRL privilege for a database or table that you create. This restriction may cause GRANT statements to fail when you might expect them to work.

For instance, suppose an ACL has no entry for PUBLIC. The following GRANT statement fails because it creates an entry for PUBLIC at the top of the ACL that does not include the DBCTRL privilege, effectively denying DBCTRL to all other entries on the list, including the owner:

```
SQL> GRANT SELECT, INSERT ON EMPLOYEES TO PUBLIC;
%RDB-E-NO_PRIV, privilege denied by database facility
```

ON SEQUENCE sequence-name
Specifies whether the GRANT statement applies to ACLs for the named sequence or sequences.

ON DATABASE ALIAS *
ON TABLE *
ON MODULE *
ON FUNCTION *
ON PROCEDURE *
ON SEQUENCE *
Specifies whether the GRANT statement applies to ACLs for all objects of the specified type. If privileges are denied for the operation on some objects, then the GRANT is aborted. However, some objects may have protection changes applied.

role-name
The name of a role, such as one created with the CREATE ROLE statement or one that can be created automatically. (If the role name exists as an operating system group or rights identifier, then Oracle Rdb will create the role automatically when you issue the GRANT statement. A role that is created automatically always has the attribute of IDENTIFIED EXTERNALLY.)

user-identifier
Specifies a user identifier that uniquely identifies each user on the system.

On OpenVMS, the user identifier consists of the standard OpenVMS user identification code (UIC), a group name, and a member name (user name). The group name is optional. The user identifier can be in either numeric or alphanumeric format. The following are all valid user identifiers that could identify the same user:

- K_JONES
- [SYSTEM3, K_JONES]
- [341,311]

In ANSI/ISO-style user identifiers, the only wildcard allowed is in the public identifier [*,*].

When Oracle Rdb creates an ANSI/ISO-style database, the creator of the database gets all privileges, and the PUBLIC entry gets no privileges.

In an ANSI/ISO-style database, you cannot use multiple user identifiers for any entry except PUBLIC.

Usage Notes

- When you grant privileges to a system user, if security checking is set to internal, then SQL automatically specifies that system user as a database user.

In effect, it is as though you issued a CREATE USER statement for the system user. See Example 4 in the Examples section.

- Prior to Oracle Rdb release 7, privileges required for data manipulation operations on global and local temporary tables were the same as those required for base tables. For example, to perform an insert into a global temporary table, a user needed SELECT+INSERT privileges at the database level.

  This requirement existed because an insert operation into a base table implicitly inserts data into the database. The privilege granted at the database level was used to filter the privileges for the table.

  However, unlike base tables, the data in temporary tables is not actually stored in the database, thus temporary tables never update the database.

  Beginning with release 7 of Oracle Rdb, only the privileges associated with the temporary table are considered when performing security validation during data manipulation operations. For example, if a user can attach to the database (requires SELECT privilege only) and is granted INSERT privilege to a global or local temporary table, then the user (or an invoker's rights stored routine) can update the temporary table. This change affects the operation of SQL*Net for Rdb, which no longer requires database manipulation privileges (INSERT, UPDATE, DELETE) for processing temporary tables.

  This relaxation of the security checking applies only to temporary tables.

Examples

Example 1 Granting SELECT Privilege to All Users for a Sequence

```
SQL> SHOW PROTECTION ON SEQUENCE EMPID
Protection on Sequence EMPID
    (IDENTIFIER=[RDB,STRAUTS],ACCESS=SELECT+SHOW+ALTER+DROP+DBCTRL)
    (IDENTIFIER=[*,*],ACCESS=NONE)
SQL> GRANT SELECT ON SEQUENCE EMPID TO PUBLIC;
SQL> SHOW PROTECTION ON SEQUENCE EMPID;
Protection on Sequence EMPID
    (IDENTIFIER=[RDB,STRAUTS],ACCESS=SELECT+SHOW+ALTER+DROP+DBCTRL)
    (IDENTIFIER=[*,*],ACCESS=SELECT)
```

Example 2 Granting INSERT ON TABLE Privilege to a Role

```
SQL> SHOW PROTECTION ON TABLE JOBS
Protection on Table JOBS
    (IDENTIFIER=[250,254],ACCESS=SELECT+INSERT+UPDATE+DELETE+SHOW+CREATE+ALTER+
```

```
      DROP+DBCTRL+OPERATOR+DBADM+REFERENCES)
    (IDENTIFIER=PUBLIC,ACCESS=SELECT+INSERT+UPDATE+DELETE+SHOW+CREATE+ALTER+DROP
      +OPERATOR+DBADM+REFERENCES)
SQL> CREATE ROLE ADMINISTRATOR;
SQL> GRANT INSERT ON TABLE JOBS TO ADMINISTRATOR AFTER [250,254];
SQL> SHOW PROTECTION ON TABLE JOBS
Protection on Table JOBS
    (IDENTIFIER=[250,254],ACCESS=SELECT+INSERT+UPDATE+DELETE+SHOW+CREATE+ALTER+
      DROP+DBCTRL+OPERATOR+DBADM+REFERENCES)
    (IDENTIFIER=ADMINISTRATOR,ACCESS=INSERT)
    (IDENTIFIER=PUBLIC,ACCESS=SELECT+INSERT+UPDATE+DELETE+SHOW+CREATE+ALTER+DROP
      +OPERATOR+DBADM+REFERENCES)
```

Example 3 Allowing All Access to a User

```
SQL> -- Allow all access to user JAIN
SQL> GRANT SELECT ON DATABASE ALIAS *   to jain;
SQL> GRANT SELECT ON TABLE *            to jain;
SQL> GRANT EXECUTE ON MODULE *          to jain;
SQL> GRANT EXECUTE ON PROCEDURE *       to jain;
SQL> GRANT EXECUTE ON FUNCTION  *       to jain;
```

Example 4 Automatically Creating a User While Granting Privileges

```
SQL> ATTACH 'FILENAME MF_PERSONNEL.RDB';
SQL> SHOW USERS
Users in database with filename mf_personnel.rdb
     tsmith
     jstuart
SQL> GRANT ALL ON DATABASE ALIAS RDB$DBHANDLE TO CDAY;
%RDB-W-META_WARN, metadata successfully updated with the reported warning
-RDMS-W-PRFCREATED, some users or roles were created
SQL> SHOW USERS
Users in database with filename mf_personnel.rdb
     tsmith
     jstuart
     cday
```

GRANT Statement: ANSI/ISO-Style

Creates or adds ANSI/ISO-style privileges to an entry of the Oracle Rdb access privilege set for a database, table, view, column, module, external routine, or sequence. At database creation time, you specify whether the database protection mechanism will be ANSI/ISO-style or ACL-style.

For more information on creating or changing the style of privileges associated with a database, see the Create Database Statement in the *Oracle Rdb7 SQL Reference Manual*.

Each entry in an ANSI/ISO-style access privilege set consists of an identifier and a list of privileges assigned to the identifier:

- Each identifier specifies a user or PUBLIC access.
- The set of privileges specifies what operations that user can perform on the database, table, column, module, external routine, or sequence.

ANSI/ISO-style privileges:

- Grant access to the creator when an object is created. Because only the creator is granted access to the newly created object, additional access must be granted explicitly.
- Support only the PUBLIC identifier as a wildcard.
- Support only user identifiers that translate to an OpenVMS user identification code (specified with the CREATE USER statement or automatically generated by the GRANT (ANSI/ISO) statement).

For ANSI/ISO-style databases, a user's privileges are a combination of all privilege sets that apply to that user. The access privilege set is not order-dependent. The user matches the entry in the access privilege set; receives whatever privileges have been granted for the database, table, column, module, external routine, or sequence; and receives the privileges defined for PUBLIC. A user without an entry in the access privilege set receives only the privileges defined for PUBLIC, which always has an entry in the access privilege set even if PUBLIC has no access to the database, table, column, module, external routine, or sequence.

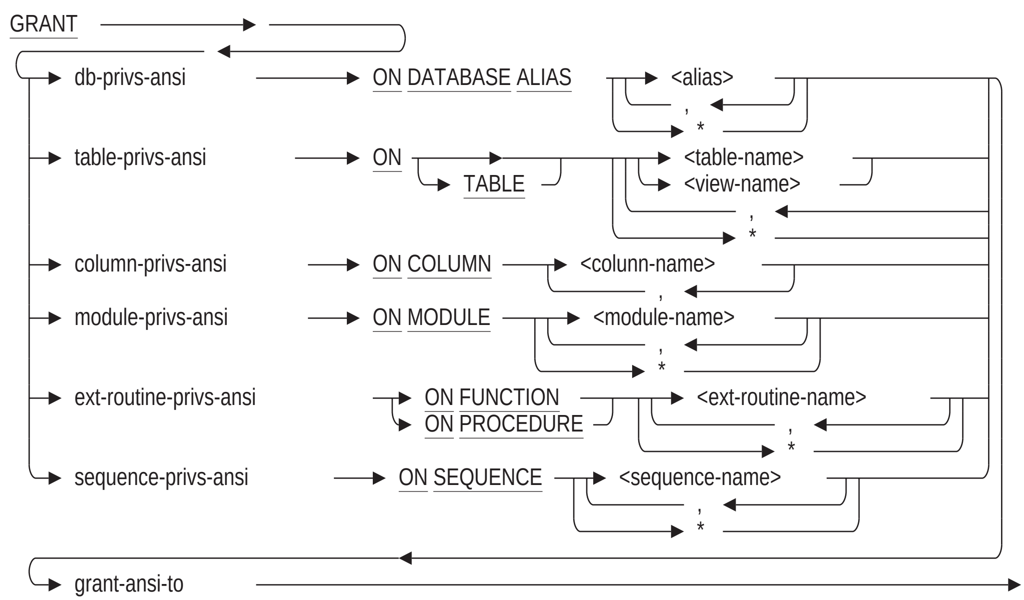
To remove privileges from, or entirely delete an entry to the Oracle Rdb access privilege set for a database, table, view, column, module, external routine, or sequence, see the REVOKE Statement: ANSI/ISO-Style in this manual and in the *Oracle Rdb7 SQL Reference Manual*.

Environment

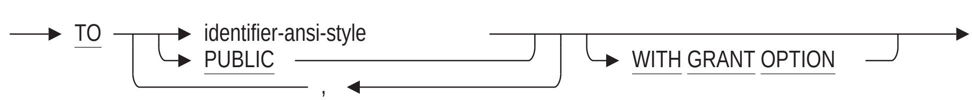
You can use the GRANT statement:

- In interactive SQL
- Embedded in host language programs to be precompiled
- As part of a nonstored procedure in a nonstored SQL module
- In dynamic SQL as a statement to be dynamically executed

Format



grant-ansi-to =



db-privs-ansi =

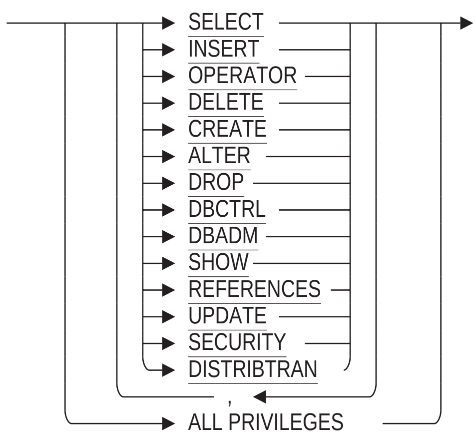
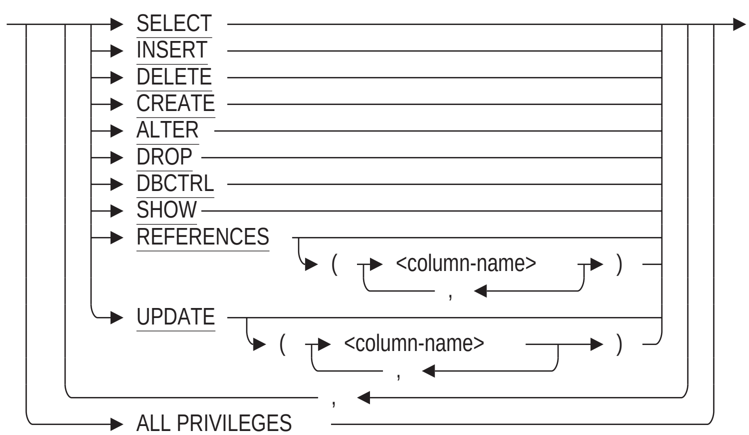
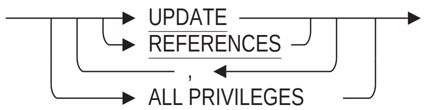


table-privs-ansi =



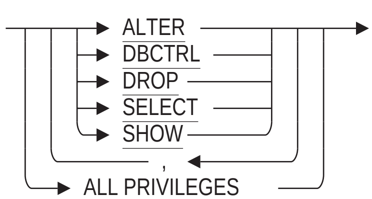
column-privs-ansi =



module-privs-ansi =

ext-routine-privs-ansi =

sequence-privs-ansi =



identifier-ansi-style =

Arguments

db-privs-ansi
table-privs-ansi
column-privs-ansi
module-privs-ansi
ext-routine-privs-ansi
sequence-privs-ansi
Specifies the list of privileges that you want to add to an existing access privilege set entry or create in a new one. The operations permitted by a given privilege keyword differ, depending on whether you granted it for a database, table, column, module, external routine, or sequence.

ON SEQUENCE sequence-name
Specifies whether the GRANT statement applies to sequences. You can specify a list of names for any form of the ON clause.

ON DATABASE ALIAS *
ON TABLE *
ON MODULE *
ON FUNCTION *
ON PROCEDURE *
ON SEQUENCE *
Specifies whether the GRANT statement applies to ACLs for all objects of the specified type. If privileges are denied for the operation on some objects, then the GRANT is aborted. However, some objects may have protection changes applied.

ALL PRIVILEGES
Specifies that SQL should grant all privileges to the specified users or roles.

role-name
The name of a role, such as one created with the CREATE ROLE statement or one that can be created automatically. (If the role name exists as an operating system group or rights identifier, then Oracle Rdb will create the role automatically when

you issue the GRANT statement. A role that is created automatically always has the attribute of IDENTIFIED EXTERNALLY.)

user-identifier

Specifies a user identifier that uniquely identifies each user on the system.

On OpenVMS, the user identifier consists of the standard OpenVMS user identification code (UIC), a group name, and a member name (user name). The group name is optional. The user identifier can be in either numeric or alphanumeric format. The following are all valid user identifiers that could identify the same user:

- K_JONES
- [SYSTEM3, K_JONES]
- [341,311]

In ANSI/ISO-style user identifiers, the only wildcard allowed is in the public identifier [*,*].

When Oracle Rdb creates an ANSI/ISO-style database, the creator of the database gets all privileges, and the PUBLIC entry gets no privileges.

In an ANSI/ISO-style database, you cannot use multiple user identifiers for any entry except PUBLIC.

Usage Notes

- A user must have SELECT privileges on a sequence to use the NEXTVAL and CURRVAL pseudocolumns.
- A user must have the WITH GRANT OPTION privilege to grant access to sequences or revoke access from sequences for other users.
- When you grant privileges to a system user, if security checking is set to internal, then SQL automatically specifies that system user as a database user. In effect, it is as though you issued a CREATE USER statement for the system user.

Examples

Example 1 Granting Privileges on a Sequence

```
SQL> GRANT ALL PRIVILEGES ON SEQUENCE EMPLOYEE_ID_GEN TO ISTEWART;
SQL> SHOW PROTECTION ON SEQUENCE EMPLOYEE_ID_GEN;
Protection on Sequence EMPLOYEE_ID_GEN
```

```
    (IDENTIFIER=[RDB,ISTEWART],ACCESS=SELECT+SHOW+ALTER+DROP+DBCTRL)
    (IDENTIFIER=[*,*],ACCESS=NONE)
SQL> GRANT SELECT ON SEQUENCE EMPLOYEE_ID_GEN TO PUBLIC;
SQL> SHOW PROTECTION ON SEQUENCE EMPLOYEE_ID_GEN;
Protection on Sequence EMPLOYEE_ID_GEN
    (IDENTIFIER=[RDB,ISTEWART],ACCESS=SELECT+SHOW+ALTER+DROP+DBCTRL)
    (IDENTIFIER=[*,*],ACCESS=SELECT)
```

GRANT Statement: Roles

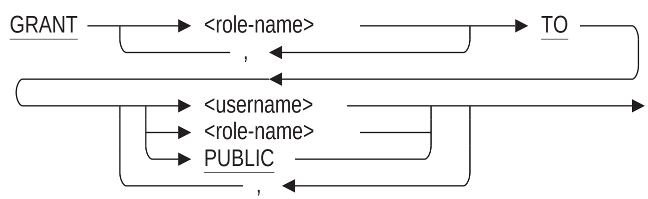
Grants a role to a user, another role, or the PUBLIC user and provides internal security checking.

Environment

You can use the GRANT statement for roles:

- In interactive SQL
- Embedded in host language programs to be precompiled
- As part of a procedure in an SQL module
- In dynamic SQL as a statement to be dynamically executed

Format



Arguments

role-name
The name of a role previously created with the CREATE ROLE statement or that can be created automatically. (If the role name exists as an operating system group or rights identifier, then Oracle Rdb will automatically create the role when you issue the GRANT statement. A role that is created automatically always has the attribute IDENTIFIED EXTERNALLY.)

TO username
TO role-name
TO PUBLIC
Specifies the user name, role name, or the PUBLIC user to which you want to grant the role. The PUBLIC user is the user name associated with all anonymous users who access the database.

(If the user name or role name exists as an operating system group or rights identifier, then Oracle Rdb will automatically create the user name or role name when you issue the GRANT statement. A role or user that is created automatically is identified externally.)

Usage Notes

- You must have the SECURITY privilege on the database to grant a role to a user or another role.
- The role being granted must not implicitly or explicitly be granted that role. This means that a role cannot be granted to itself.

Examples

Example 1 Granting and Revoking Roles

```
SQL> -- Create three users and two roles. Oracle Rdb automatically
SQL> -- generates users and roles if they are identified externally.
SQL> CREATE USER ABLOWNEY IDENTIFIED EXTERNALLY;
SQL> CREATE USER BGREMBO IDENTIFIED EXTERNALLY;
SQL> CREATE USER LWARD IDENTIFIED EXTERNALLY;
SQL> CREATE ROLE SALES_MANAGER IDENTIFIED EXTERNALLY;
SQL> CREATE ROLE DIVISION_MANAGER IDENTIFIED EXTERNALLY;
SQL> -- Grant the SALES_MANAGER role to users ABLOWNEY,
SQL> -- BGREMBO, and to the DIVISION MANAGER ROLE.
SQL> GRANT SALES_MANAGER TO ABLOWNEY, BGREMBO, DIVISION_MANAGER;
SQL> -- Grant the DIVISION_MANAGER role to LWARD. LWARD now
SQL> -- has both the SALES_MANAGER and DIVISION_MANAGER roles.
SQL> GRANT DIVISION_MANAGER TO LWARD;
SQL> -- Revoke the DIVISION_MANAGER role from LWARD. He has
SQL> -- left the company.
SQL> REVOKE DIVISION_MANAGER FROM LWARD;
SQL> -- Grant the DIVISION_MANAGER role to BGREMBO.  She
SQL> -- has been promoted to division manager.
SQL> GRANT DIVISION_MANAGER TO BGREMBO;
```

IMPORT Statement

Creates an Oracle Rdb database from an interchange .rbr file.

You use the IMPORT statement with the EXPORT statement to make changes to Oracle Rdb databases that cannot be made any other way. The EXPORT statement unloads a database to an .rbr file. The IMPORT statement re-creates the database with changes that are both allowed and not allowed through ALTER statements. The IMPORT statement lets you:

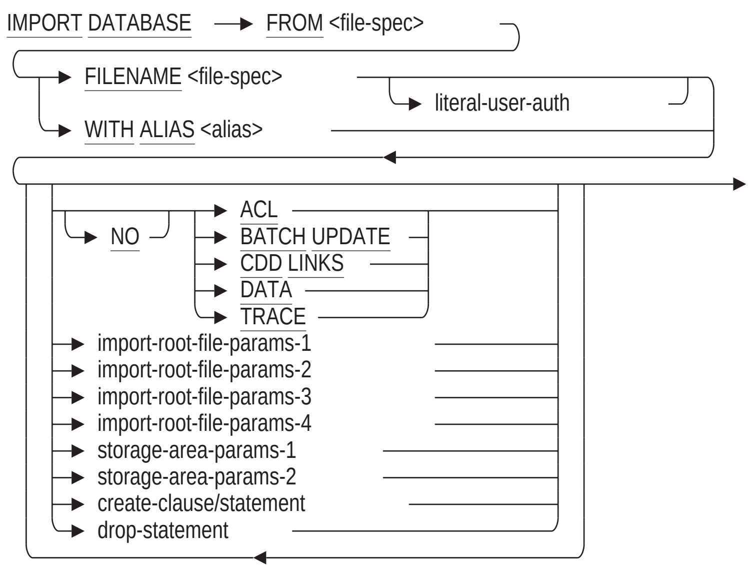
- Convert from a single-file to a multifile database, and vice versa.
- Change database root file parameters that you cannot change with the ALTER DATABASE statement:
  - COLLATING SEQUENCE
  - SEGMENTED STRING STORAGE AREA
  - PROTECTION IS ANSI / ACL
  - DEFAULT STORAGE AREA
- Change storage area parameters that you cannot change with the ALTER DATABASE statement:
  - PAGE SIZE
  - PAGE FORMAT
  - THRESHOLDS
  - INTERVAL
  - SNAPSHOT FILENAME
- Reload tables with existing rows to take advantage of newly created hashed indexes.
- Reload tables to take advantage of new or changed storage maps.
- Move a database to another directory or disk structure. However, if moving a database is the only change you need to make, it is more efficient to use the RMU Backup and RMU Restore commands.
- Create an empty target database that uses the same data definitions as a source database by copying the metadata, but not the data, to the target.

If you use the NO DATA option, the IMPORT statement creates an Oracle Rdb database whose metadata is identical to that found in the source database used by the EXPORT statement, but the duplicate database contains no data. The NO DATA option is not compatible with the repository databases.

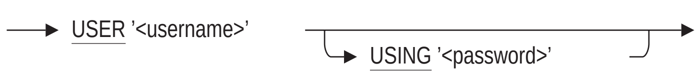
Environment

You can use the IMPORT statement in interactive SQL only.

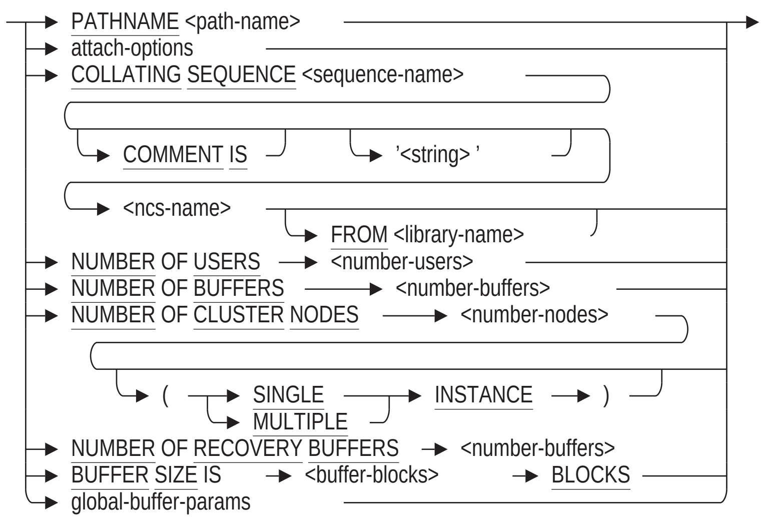
Format



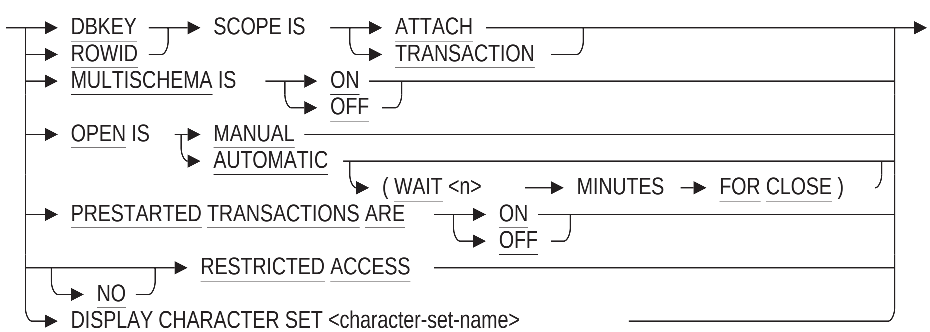
literal-user-auth =



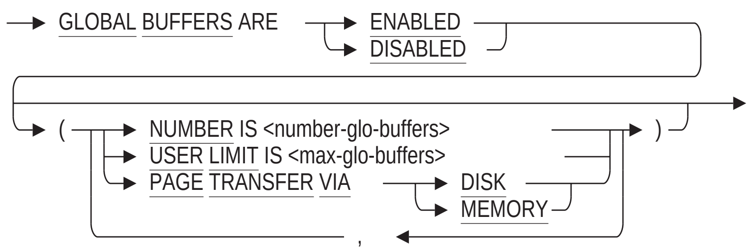
import-root-file-params-1 =



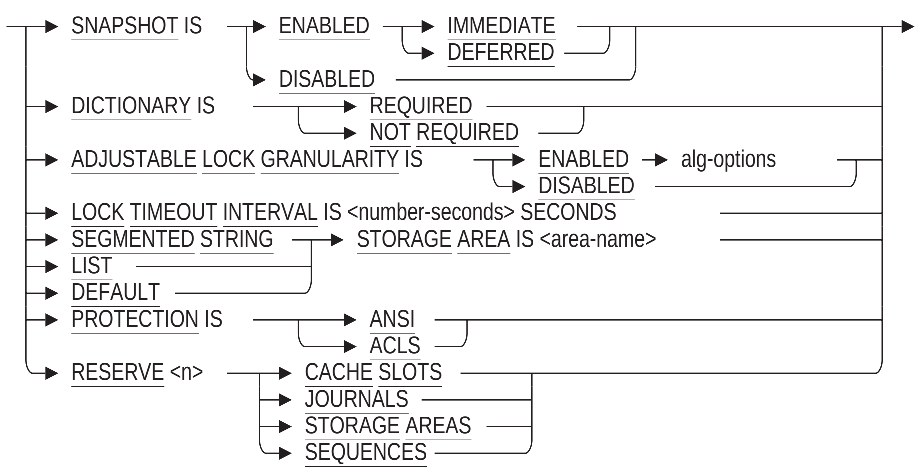
attach-options =



global-buffer-params=

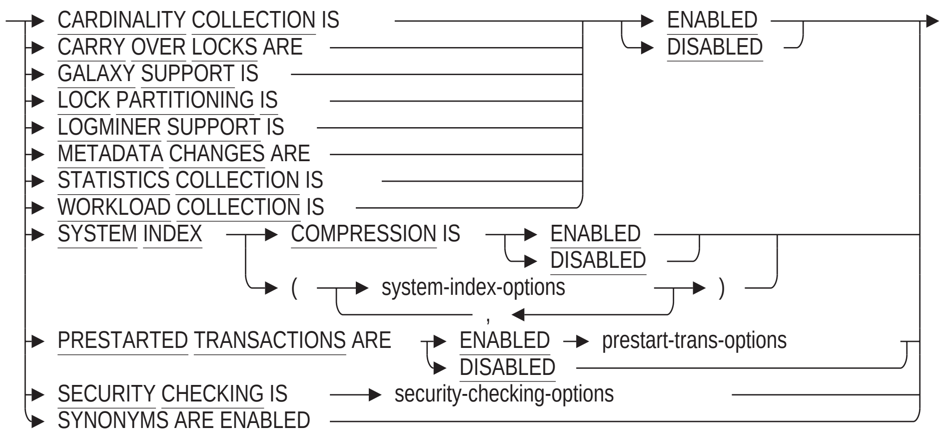


import-root-file-params-2 =

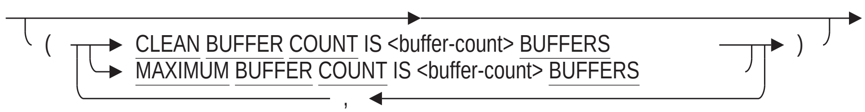


alg-options =

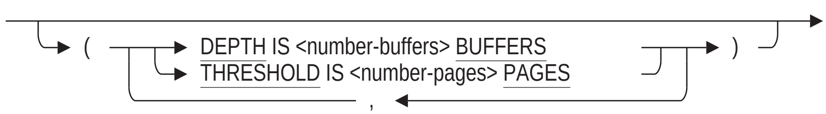
import-root-file-params-3 =



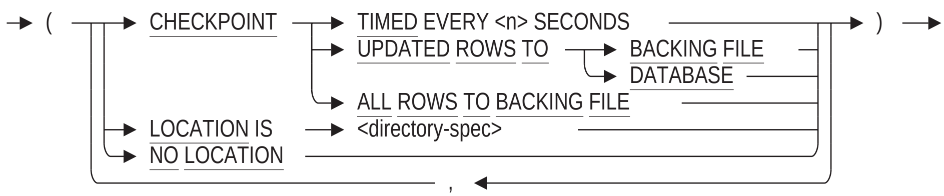
asynch-bat-wr-options =



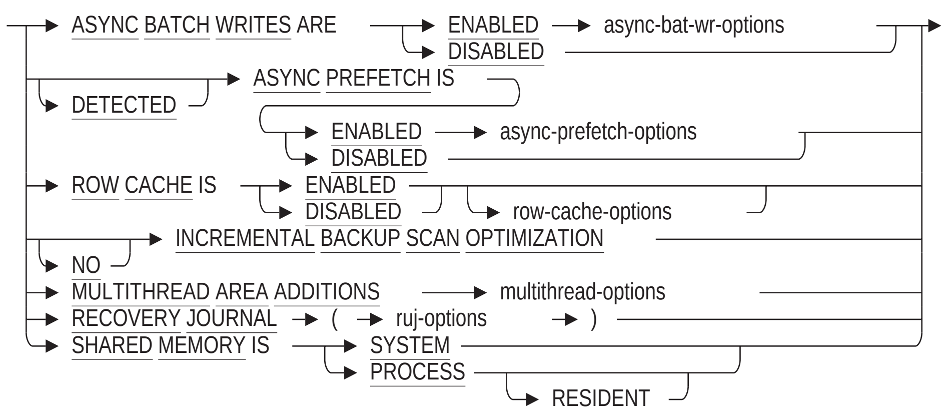
async-prefetch-options =



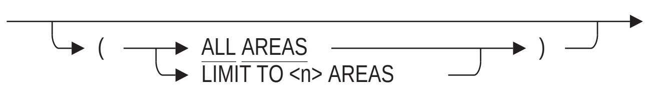
row-cache-options =



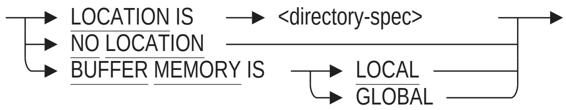
import-root-file-params-4 =



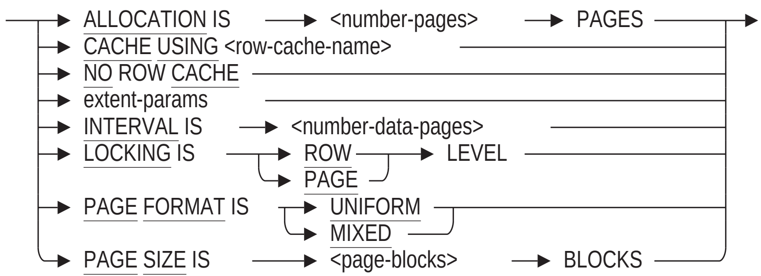
multithread-options =



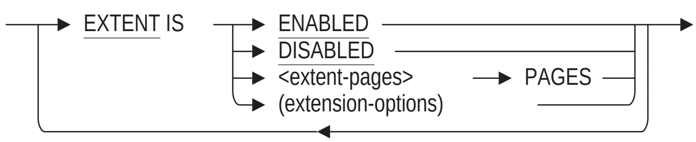
ruj-options =



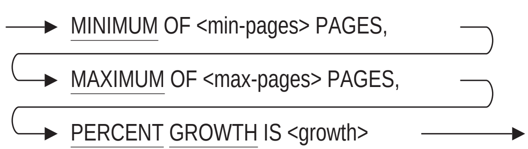
storage-area-params-1 =



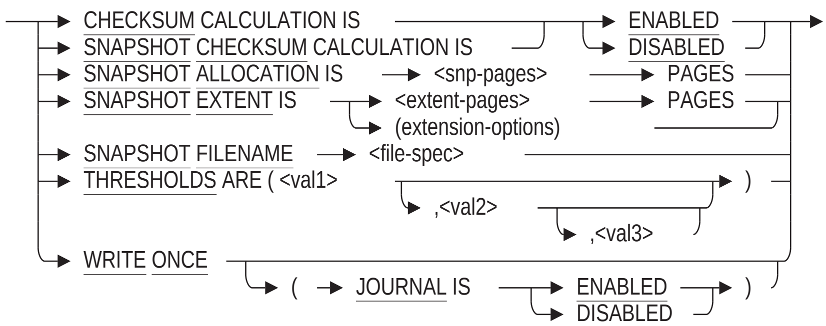
extent-params =



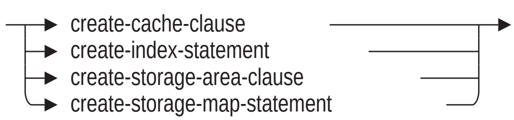
extension-options =



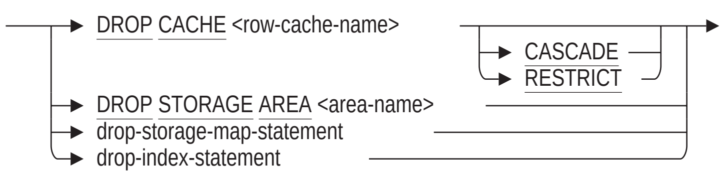
storage-area-params-2 =



create-clause/statement =



drop-statement =



Arguments

RESERVE n CACHE SLOTS

Specifies the number of row caches for which slots are reserved in the database.

You can use the RESERVE n CACHE SLOTS clause to reserve slots in the database root file for future use by the ADD CACHE clause.

Row caches can be added only if row cache slots are available. Slots become available after you issue a DROP CACHE clause or a RESERVE n CACHE SLOTS clause.

You cannot decrease the number of reserved slots for row caches after you issue the RESERVE clause. If you reserve 10 slots and later reserve 5 slots, you have a total of 15 reserved slots for row caches.

Reserving row cache slots is an offline operation (requiring exclusive database access).

RESERVE n SEQUENCES

Specifies the number of sequences for which slots are reserved in the database. You can use the RESERVE n SEQUENCES clause to reserve slots in the database root file for future use by the CREATE SEQUENCE statement. Sequences can be added only if sequence slots are available. Additional slots become available after a DROP SEQUENCE statement or a RESERVE n SEQUENCES clause of the ALTER DATABASE statement is executed.

The number of reserved slots for sequences cannot be decreased. If you reserve 32 slots and later reserve another 32 slots, you have a total of 64 reserved slots for sequences.

If you do not specify the RESERVE n SEQUENCES clause, the default number of sequence slots is 32.

RECOVERY JOURNAL (LOCATION IS directory-spec)

Specifies the location, including device and directory, in which the recovery-unit journal (.ruj) file is written. Do not include network node names, file names (including file type or version on OpenVMS), or process-concealed logical names.

If this clause is omitted, the default directory location is the current device the database root file uses and the special directory [RDM$RUJ]. You can use the RDMS$RUJ logical name to override this clause.

RECOVERY JOURNAL (NO LOCATION)

Removes a location previously defined by a RECOVERY JOURNAL LOCATION IS clause. This causes the recovery journal to revert to the default location.

RECOVERY JOURNAL (BUFFER MEMORY IS LOCAL)
RECOVERY JOURNAL (BUFFER MEMORY IS GLOBAL)

Specifies whether RUJ buffers will be allocated in global or local memory.

The RUJ buffers used by each process are normally allocated in local virtual memory. With the introduction of row caching, these buffers now can be assigned to a shared global section (global memory) on OpenVMS, so that the recovery process can process this in-memory buffer and possibly avoid disk access.

You can define this buffer memory to be global to improve row caching performance for recovery. If row caching is disabled, then buffer memory is always local.

ROW CACHE IS ENABLED
ROW CACHE IS DISABLED
Specifies whether or not the row caching feature is enabled.

Enabling row caching does not affect database operations until a cache is created and assigned to one or more storage areas.

When row caching is disabled, all previously created and assigned caches remain and will be available if row caching is enabled again.

To enable or disable row caching, you must have exclusive access to the database (it is an offline operation).

CHECKPOINT TIMED EVERY n SECONDS
Specifies the frequency with which the row cache server (RCS) process checkpoints the contents of the row caches back to disk. The RCS process does not use the checkpoint frequency options of the FAST COMMIT clause.

The frequency of RCS checkpointing is important in determining how much of an after-image journal file must be read during a recovery operation following a node failure. It also affects the frequency with which marked records get flushed back to the database, for those row caches that checkpoint to the database. The default is every 15 minutes (900 seconds).

CHECKPOINT ALL ROWS TO BACKING FILE
CHECKPOINT UPDATED ROWS TO DATABASE
CHECKPOINT UPDATED ROWS TO BACKING FILE
Specifies the default source and target during checkpoint operations for all row caches. If ALL ROWS is specified, then the source records written during each checkpoint operation are both the modified and the unmodified rows in a row cache. If UPDATED ROWS is specified, then just the modified rows in a row cache are checkpointed each time.

If the target of the checkpoint operation is BACKING FILE, then the RCS process writes the source row cache entries to the backing (.rdc) files. The row cache LOCATION, ALLOCATION, and EXTENT clauses are used to create the backing files. Upon recovery from a node failure, the database recovery process is able to repopulate the row caches in memory from the rows found in the backing files.

If the target is DATABASE, then updated row caches entries are written back to the database. The row cache LOCATION, ALLOCATION, and EXTENT clauses are ignored. Upon recovery from a node failure, the database recovery process has no data on the contents of the row cache. Therefore, it does not repopulate the row caches in memory.

The CHECKPOINT clause of the CREATE CACHE, ADD CACHE, or ALTER CACHE clause overrides this database-level CHECKPOINT clause.

SECURITY CHECKING

Traditionally, Oracle Rdb has performed security checking using the operating system security layer (for example, the UIC and rights identifiers of the OpenVMS operating system).

The access control list (ACL) information stored in the database contains a granted privilege mask and a set of users represented by a unique integer (for example, a UIC).

There are three modes of security checking:

- SECURITY CHECKING IS EXTERNAL

  This is the default. External security checking recognizes database users (such as those created with the SQL CREATE USER statement) as operating system user identification codes (UICs) and roles as special rights identifiers or groups.

- SECURITY CHECKING IS INTERNAL (ACCOUNT CHECK IS ENABLED)

  The ACCOUNT CHECK clause ensures that Oracle Rdb validates the current database user with the user name (such as one defined with an SQL CREATE USER statement) stored in the database. This prevents different users with the same name from accessing the database. Therefore, this clause might prevent a breach in security.

Note: The ACCOUNT CHECK IS ENABLED clause on OpenVMS forces the user session to have the same user name and UIC as recorded in the database.

If you specify the ACCOUNT CHECK IS DISABLED clause, then a user with a matching UIC (also called a profile-id) is considered the same as the user even if his or her user name is different. This allows support for multiple OpenVMS users with the same UIC.

- SECURITY CHECKING IS INTERNAL (ACCOUNT CHECK IS DISABLED)

  This syntax means that the assigned SID is ignored during database attach.

CACHE USING row-cache-name

Specifies that the named row cache is the default physical row cache for all storage areas in the database. All rows stored in each storage area are cached, regardless of whether they consist of table data, segmented string data, or special rows such as index nodes.

You must either add the specified cache before completing the ALTER DATABASE statement, or it must already exist.

Alter the database and storage area to assign a new physical area row cache that overrides the database default physical area row cache. Only one physical area row cache is allowed for each storage area.

You can have multiple row caches that contain rows for a single storage area by defining logical area row caches, where the row cache name matches the name of a table or index.

If you do not specify the CACHE USING clause or the NO ROW CACHE clause, then the NO ROW CACHE clause is the default.

NO ROW CACHE

Specifies that the database default is to not assign a row cache to all storage areas in the database. You cannot specify the NO ROW CACHE clause if you specify the CACHE USING clause.

Alter the storage area and name a row cache to override the database default. Only one row cache is allowed for each storage area.

If you do not specify the NO ROW CACHE clause or the CACHE USING clause, then the NO ROW CACHE clause is the default.

create-cache-clause

See the CREATE CACHE Clause for a description of this clause and its arguments.

DROP CACHE row-cache-name CASCADE
DROP CACHE row-cache-name RESTRICT

Deletes the specified row cache from the database. If the mode is RESTRICT, then an exception is raised if the row cache is assigned to a storage area. If the mode is CASCADE, then the row cache is removed from all referencing storage areas.

The default is RESTRICT if no mode is specified.

Usage Notes

- By default, a transaction that reserves a table for EXCLUSIVE access does not reserve the LIST (segmented string) area for exclusive access. Because the LIST area is usually shared by many tables, SHARED access is assumed by default to permit updates to the other tables.

  This means that when you run an Oracle RMU load operation or an application updates a table reserved for EXCLUSIVE access, you might notice that the snapshot storage area (.snp) grows. This is because of the I/O to the LIST area that is performed by default when SHARED WRITE mode is in use.

  However, if you attach to the database using an SQL ATTACH statement and you specify the RESTRICTED ACCESS clause, then all storage areas are accessed in EXCLUSIVE mode. Use this clause to eliminate the snapshot I/O and related overhead if you are performing a lot of I/O to the LIST storage areas (for example, when you are restructuring the database, or dropping a large table containing LIST OF BYTE VARYING columns and data).

  Because RESTRICTED ACCESS is the default for the SQL IMPORT statement, there is reduced overhead during an import of LIST data.

Examples

Example 1 Reserving Sequence Slots During an Import Operation

```
SQL> IMPORT DATABASE FROM MF_PERSONNEL.RBR
cont> FILENAME 'mf_personnel.rdb'
cont> RESERVE 64 SEQUENCES;
.
.
.
Unused Sequences were 32 now are 64
IMPORTing STORAGE AREA: RDB$SYSTEM
```

```
IMPORTing STORAGE AREA: DEPARTMENTS
IMPORTing STORAGE AREA: EMPIDS_LOW
```

INSERT Statement

Adds a new row, or a number of rows, to a table or view. You can also use the INSERT statement with a cursor to assign values to the segments in a column of the LIST OF BYTE VARYING data type.

Before you assign values to the segments in a column of the LIST OF BYTE VARYING data type, you must first assign a value to one or more other columns in the same row. To do this, use a positioned insert. A **positioned insert** is an INSERT statement that specifies an insert-only table cursor. This type of INSERT statement sets up the proper row context for subsequent list cursors to assign values to list segments.

You can specify the name of a static cursor, a dynamic cursor, or an extended dynamic cursor in a positioned insert. If you specify a static cursor name, that cursor name must also be specified in a DECLARE CURSOR statement within the same module. See the description of the DECLARE CURSOR statement in the *Oracle Rdb7 SQL Reference Manual* for more information on static, dynamic, and extended dynamic cursors.

When you use an INSERT statement to assign values to list segments:

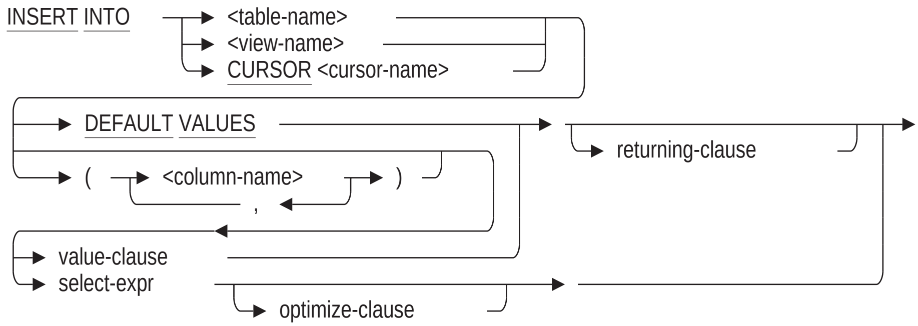
- The current transaction must not be read-only.
- You cannot specify a cursor name that refers to an update table cursor.
- Your cursor must specify an intermediate table.
- The value that you assign is appended to the end of the list.

Environment

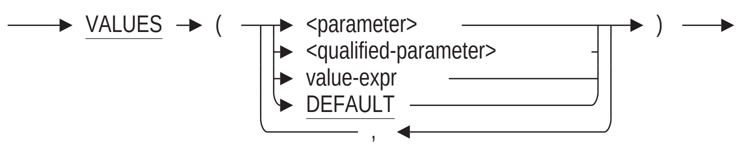
You can use the INSERT statement:

- In interactive SQL
- Embedded in host language programs to be precompiled
- As part of a procedure in an SQL module
- In dynamic SQL as a statement to be dynamically executed

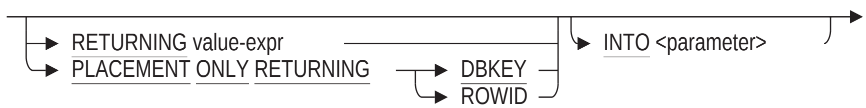
Format



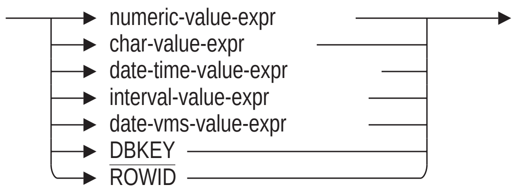
value-clause =



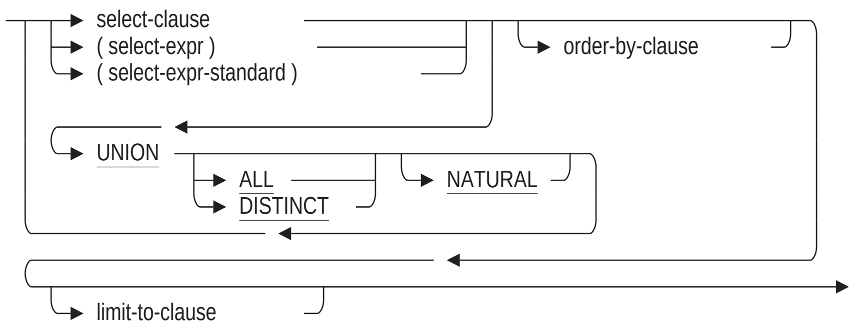
returning-clause =



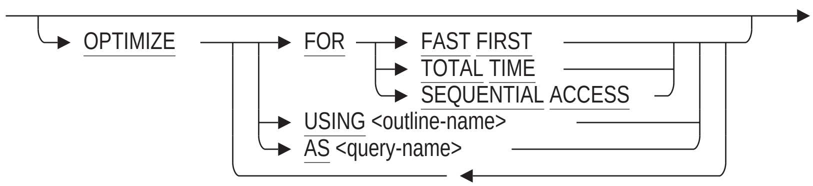
value-expr =



select-expr =



optimize-clause =



Arguments

DEFAULT VALUES

Specifies that every column in the table is assigned the default value (or NULL, if the column has no default value).

DEFAULT

Forces the named column to assume the default value defined for that column.

Usage Notes

The default-value clause that previously appeared (before release 7.1) in the format for the INSERT statement is now obsolete. All functions previously listed under the default-value clause now appear under the value-expr clause.

Examples

Example 1 Inserting Default Values for Selected Columns

```
SQL> INSERT INTO DEPARTMENTS
cont> (DEPARTMENT_CODE, DEPARTMENT_NAME, BUDGET_ACTUAL)
cont> VALUES
cont> ('RECR','Recreation', DEFAULT);
1 row inserted
SQL> SELECT * FROM DEPARTMENTS WHERE DEPARTMENT_CODE='RECR';
 DEPARTMENT_CODE   DEPARTMENT_NAME                  MANAGER_ID
   BUDGET_PROJECTED   BUDGET_ACTUAL
 RECR              Recreation                       NULL
               NULL            NULL
1 row selected
```

Example 2 Inserting a Row of All Default Values into a Table

```
SQL> INSERT INTO CANDIDATES
cont> DEFAULT VALUES;
1 row inserted
SQL> SELECT * FROM CANDIDATES
cont> WHERE LAST_NAME IS NULL;
 LAST_NAME        FIRST_NAME   MIDDLE_INITIAL
   CANDIDATE_STATUS
      RESUME
 NULL             NULL         NULL
   NULL
   >>
```

```
   >>
   >>
      NULL
1 row selected
```

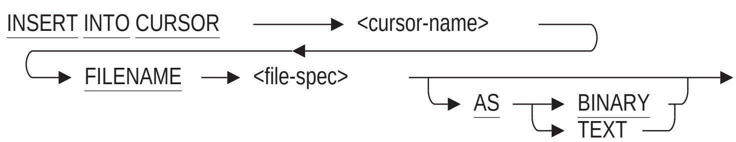
INSERT from FILENAME Statement

Allows you to load a column of the LIST OF BYTE VARYING data type from a text or binary file without needing to use special application code. The specified file is opened and each row is read and stored in the LIST OF BYTE VARYING column specified by the list cursor.

Environment

You can use the INSERT from FILENAME statement in interactive SQL only.

Format



Arguments

INSERT INTO CURSOR cursor-name
The name of the target list cursor to which you want to add a list segment.

FILENAME filespec
The specification for the file that you want to load into the LIST OF BYTE VARYING column.

AS BINARY
AS TEXT
Specifies whether the file specified with the FILENAME clause contains binary or text data.

Usage Notes

- When you use an INSERT from FILENAME statement to assign values to list segments:
  - The current transaction must be read / write.

– Your cursor must specify an insert-only list cursor.

Examples

Example 1 Adding a New Row Using Data from a Text File

```
SQL> -- Declare a table cursor.
SQL> DECLARE TABLE_CURSOR
cont> INSERT ONLY TABLE CURSOR
cont> FOR SELECT * FROM RESUMES;
SQL> -- Open table cursor and insert values.
SQL> OPEN TABLE_CURSOR;
cont> INSERT INTO CURSOR TABLE_CURSOR
cont> VALUES ('10065', NULL);
1 row inserted
SQL> -- Declare a list cursor.
SQL> DECLARE LIST_CURSOR
cont> INSERT ONLY LIST CURSOR
cont> FOR SELECT RESUME WHERE CURRENT OF TABLE_CURSOR;
SQL --Open list cursor.
SQL> OPEN LIST_CURSOR;
SQL> --Load text from file into LIST OF BYTE VARYING column.
SQL> INSERT INTO CURSOR LIST_CURSOR
cont> FILENAME 'resume_10065.sql' AS TEXT;
SQL> CLOSE LIST_CURSOR;
SQL> CLOSE TABLE_CURSOR;
SQL> COMMIT;
```

ITERATE Control Statement

Causes the current iteration of the loop to abort and either the next iteration to start or the loop to terminate; depending on the termination conditions.

Environment

You can use the ITERATE control statement in a compound statement of a multistatement procedure:

- In interactive SQL
- Embedded in host language programs to be precompiled
- As part of a procedure in an SQL module
- In dynamic SQL as a statement to be dynamically executed

Format

Usage Notes

- The statement label must be for an active iterative loop statement. Iterative loop statements include LOOP, FOR cursor loop, FOR counted loop, WHILE, and REPEAT statements. An exception is raised if the specified label is unknown, not active, or is not a label for an iterative statement.
- If the statement label is omitted, then the innermost statement is used by default. An exception is raised if there is no active iterative statement.

Examples

Example 1 Using the ITERATE Control Statement

The following example shows the ITERATE control statement being used to prematurely complete the processing of the current row in a FOR cursor loop:

```
SQL> BEGIN
cont>   FOR :ord AS TABLE CURSOR ord_cursor
```

```
cont>   AS SELECT * FROM orders WHERE customer_id = :cid
cont>   DO
cont>      IF stock_count (:ord.product_id, :ord.quantity) IS NULL THEN
cont>           ITERATE;
cont>      END IF;
cont>      -- transfer stock to this order
cont>      UPDATE stock SET on_hand = on_hand - :ord.quantity
cont>       WHERE product_id = :ord.product_id;
cont>      UPDATE orders SET :ord.available = :ord.quantity
cont>       WHERE CURRENT OF ord_cursor;
cont>   END FOR;
cont> END;
```

LEAVE Control Statement

Unconditionally ends execution within a compound statement block or a LOOP statement but resumes execution on any SQL statement that immediately follows the exited statement.

Environment

You can use the LEAVE control statement in a compound statement of a multistatement procedure:

- In interactive SQL
- Embedded in host language programs to be precompiled
- As part of a procedure in an SQL module
- In dynamic SQL as a statement to be dynamically executed

Format

Arguments

statement-label
Names the label assigned to a compound statement or a LOOP statement in a multistatement procedure block.

Usage Notes

- If the statement-label is omitted, then the LEAVE statement leaves the currently active loop statement (WHILE, LOOP, REPEAT, FOR cursor loop, FOR counted loop); otherwise, it leaves the current labeled statement. If there is no active loop or labeled statement, then the current procedure is terminated.
- Do not use the LEAVE statement to leave SQL functions. A function must have a return result. You will receive a run-time error if you attempt to terminate a function with the LEAVE statement. Use the RETURN statement instead.

Examples

Example 1 Ending Execution of a Compound Statement

```
PROCEDURE SAMPLE (IN :ID MONEY);
BEGIN
DECLARE: AMOUNT MONEY
         (SELECT TOTAL_AMOUNT FROM M_TABLE);
LOOP
   IF :AMOUNT IS NULL THEN
      LEAVE;
   END IF;
   .
   .
   .
   SET :AMOUNT =:AMOUNT-100.00;
   IF :AMOUNT <0.00 THEN
       LEAVE;
   END IF;
END LOOP;
END;
```

LOCK TABLE Statement

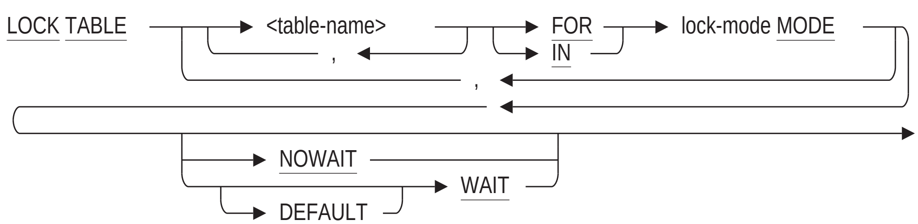
Specifies a list of tables to be readied in a given lock mode and added to the list of reserved tables for the current transaction. If a view is specified, then the base tables referenced by the view are locked in the specified lock mode.

Environment

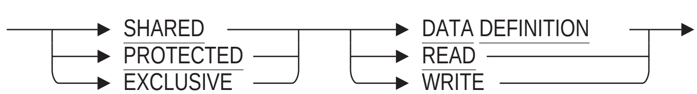
You can use the LOCK TABLE statement in a compound statement of a multistatement procedure:

- In interactive SQL
- Embedded in host language programs to be precompiled
- As part of a procedure in an SQL module
- In dynamic SQL as a statement to be dynamically executed

Format



lock-mode =



Arguments

table-name
The names of one or more tables or views currently existing in the database that you want to lock and reserve. You can specify tables created as GLOBAL or LOCAL TEMPORARY TABLES, but they will be ignored because these types of tables do not contain shared data and so are never locked. You can specify tables from multiple databases by using the alias name as a prefix to the table name. If you do not specify an alias, then the default alias is used.

IN lock-mode MODE
FOR lock-mode MODE
Specifies the lock mode to be used for the specified tables and views. The IN and FOR keywords are synonymous. A table lock mode can be promoted, but cannot be demoted. For example, you can promote a SHARED READ lock to SHARED WRITE, but you cannot demote a SHARED WRITE mode to a SHARED READ mode. See the Usage Notes for information on how the LOCK TABLE statement interacts with the SET TRANSACTION and DECLARE TRANSACTION statements.

SHARED
PROTECTED
EXCLUSIVE
See the SET TRANSACTION statement in the *Oracle Rdb7 SQL Reference Manual* for a description of these arguments.

DATA DEFINITION
READ
WRITE
See the SET TRANSACTION statement in the *Oracle Rdb7 SQL Reference Manual* for a description of these arguments.

WAIT
NOWAIT
DEFAULT WAIT
Specifies what the LOCK TABLE statement does when it encounters a locked table. If you specify WAIT, the statement waits for other transactions to complete and then proceeds. If you specify NOWAIT, your transaction returns an error message when it encounters a locked table. If you specify DEFAULT WAIT, then the lock mode specified for the current transaction is used. If you specify a different lock mode than was specified for the transaction, the mode you specify with the LOCK TABLE statement takes precedence, unless the table is already reserved.

The WAIT clause is the default.

Usage Notes

- The LOCK TABLE statement has a definite advantage over the SET TRANSACTION RESERVING clause. It allows tables to be locked at modes other than SHARED READ when the table access is not determined until run time. For example, complex or dynamic applications often do not know the names of tables that will be accessed at the time a transaction is started. The LOCK TABLE statement allows those applications to start a transaction and add tables later, as they become known.
- If you start a transaction with a SET TRANSACTION or DECLARE TRANSACTION statement that includes the RESERVING clause, then all tables referenced during that transaction must have been specified in the reserving list of that transaction or subsequently with a LOCK TABLE statement. Exceptions to this rule are temporary tables and tables that are referenced by constraints and triggers. These tables are automatically reserved according to their access characteristics. For example, constraints require read access, triggers may require write access, and temporary tables require no special locking.
- If you start a transaction without specifying a list of reserved tables, then you can reference any tables during the transaction. By default, they will be accessed for SHARED READ or SHARED WRITE depending on the type of access statement issued. You can use the LOCK TABLE statement to adjust the default locking behavior as needed by the transaction.
- When you use multiple LOCK TABLE statements in a transaction, the tables can be reserved in any order and at any time, as you desire. However, this may lead to deadlocks in concurrent environments. Careful design can eliminate or minimize this problem. (Contrast this with the behavior seen when you use the SET TRANSACTION statement with the RESERVING clause. In this case, the tables are reserved using the order specified by the RDB$RELATION_ID column of the RDB$RELATION system relation so that a consistent ordering is used across every application. This avoids or eliminates deadlocks during table reservation.)
- If you issue a LOCK TABLE statement when no transaction is active, then a default transaction is started implicitly.
- The locks placed on tables by the LOCK TABLE statement are released when the transaction is terminated with a COMMIT, ROLLBACK, or DISCONNECT statement.
- You cannot demote a lock mode.

Examples

Example 1 Locking a Table in READ MODE

```
SQL> LOCK TABLE EMPLOYEES IN PROTECTED READ MODE NOWAIT;
```

Example 2 Locking Two Tables in Different Modes

```
SQL> LOCK TABLE DB1.JOB_HISTORY IN SHARED WRITE MODE,
cont>      dB2.SALARY_HISTORY IN EXCLUSIVE WRITE MODE;
```

REPEAT Control Statement

Repetitively executes one or more SQL statements in a compound loop until an end condition is met.

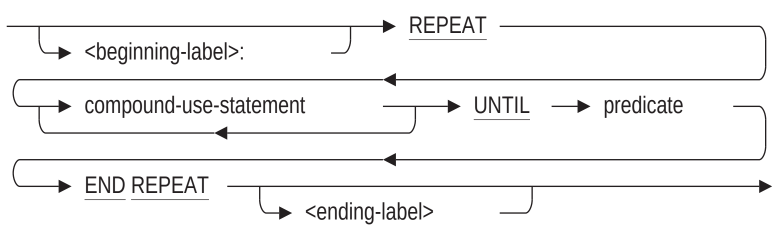
Environment

You can use the REPEAT control statement in a compound statement of a multistatement procedure:

- In interactive SQL
- Embedded in host language programs to be precompiled
- As part of a procedure in an SQL module
- In dynamic SQL as a statement to be dynamically executed

Format

repeat-statement=



Arguments

beginning-label

Assigns a name to the REPEAT statement. A beginning label used with the LEAVE statement lets you perform a controlled exit from a repeat loop. A named repeat loop is called a **labeled repeat statement**. If you include an ending label, it must be identical to its corresponding beginning label. A beginning label must be unique within the procedure in which the label is contained.

REPEAT compound-use-statement
Repeatedly executes a block of SQL statements until an end condition is met, as specified by the UNTIL predicate clause.

UNTIL predicate
Specifies a condition that controls how many times SQL can execute the statements embedded within its REPEAT ... UNTIL block (collectively referred to as its compound statement). SQL executes the compound statement once and then evaluates the UNTIL condition. If it evaluates to false or NULL (unknown) and does not encounter an error exception, SQL executes the compound statement again. Each time the search condition evaluates to false or NULL, the REPEAT statement executes the compound statement. If the UNTIL condition evaluates to true, SQL bypasses the compound statement and passes control to the statement after the END REPEAT statement.

END REPEAT ending-label
Marks the end of a control loop. If you choose to include the optional ending label, it must match exactly its corresponding beginning label. An ending label must be unique within the procedure in which the label is contained.

The optional ending-label argument makes multistatement procedures easier to read, especially in complex multistatement procedure blocks.

Usage Notes

The loop body is executed at least once for a REPEAT statement.

Examples

Example 1 Using a REPEAT Statement to List Files in the Current Directory

```
SQL> SET VERIFY;
SQL> ATTACH 'FILE SCRATCH';
SQL> CREATE DOMAIN file_name  VARCHAR(255);
SQL>  CREATE PROCEDURE find_file
cont>    (IN :FILESPEC file_name BY DESCRIPTOR,
cont>     INOUT :RESULTANT_FILESPEC file_name BY DESCRIPTOR,
cont>     INOUT :CONTEXT INTEGER BY REFERENCE);
cont> EXTERNAL NAME LIB$FIND_FILE
cont> LOCATION 'SYS$LIBRARY:LIBRTL.EXE'
cont> LANGUAGE GENERAL
cont> PARAMETER STYLE GENERAL
cont> COMMENT IS
cont>    'DCL HELP: LIB$FIND_FILE '
```

```
cont>     / 'The Find File routine is called with a wildcard  file'
cont>      / 'specification for  which   it   searches.    LIB$FIND_FILE '
cont>    /  'returns   all   file specifications that satisfy that'
cont>   /' wildcard file specification.';
SQL> CREATE PROCEDURE Find_file_end
cont>    (IN :CONTEXT INTEGER BY REFERENCE);
cont> EXTERNAL
cont>    NAME LIB$FIND_FILE_END
cont>     LOCATION 'SYS$LIBRARY:LIBRTL.EXE'
cont>     LANGUAGE GENERAL
cont>    PARAMETER STYLE GENERAL
cont> COMMENT IS
cont>   'DCL HELP: LIB$FIND_FILE_END '
cont>    / 'The End of Find File routine is called once'
cont>    / 'after each  sequence  of '
cont>    / 'calls to LIB$FIND_FILE. LIB$FIND_FILE_END deallocates'
cont>    / 'any saved Record Management Service (RMS) context and'
cont>    / 'deallocates the virtual memory used to hold the'
cont>    / 'allocated context block.';
SQL> SET FLAGS 'TRACE';
SQL> BEGIN
cont> -- This procedure performs a call to an external
cont> -- routine to list files located in the current
cont> -- default directory
cont> DECLARE :done, :context integer = 0;
cont> DECLARE :search_string FILE_NAME = '*.SQL';
cont> DECLARE :file_spec FILE_NAME;
cont> REPEAT
cont>     -- Ask the OpenVMS routine for the next name
cont>     CALL find_file (:search_string, :file_spec, :context);
cont>     IF POSITION ('*' in :file_spec) = 0
cont>        AND POSITION ('%' in :file_spec) = 0
cont>        AND POSITION ('...' in :file_spec) = 0
cont>     THEN
cont>         -- Display the name (there are no wildcards)
cont>         TRACE :file_spec;
cont>     ELSE
cont>        SET :done = 1;
cont>    END IF;
cont>   -- Exit when we have no more file names
cont>   UNTIL :done = 1
cont> END REPEAT;
cont> -- Clean up search context
cont> CALL find_file_end (:context);
cont> END;
```

```
~Xt: RDBVMS:[USER.V80]CREATE_ROLES.SQL;1
~Xt: RDBVMS:[USER.V80]TEST.SQL;1
SQL>
```

REVOKE Statement

Removes privileges from or entirely deletes an entry to the Oracle Rdb access privilege set (called the **access control list**) for a database, table, column, module, external routine, or sequence. Each entry in an access control list (ACL) consists of an identifier and a list of privileges assigned to the identifier.

- Each identifier specifies a user or a set of users.
- The list of privileges specifies which operations that user or user group can perform on the database, table, column, module, external routine, or sequence.

When a user tries to perform an operation on a database, SQL reads the associated ACL from top to bottom, comparing the identifier of the user with each entry. As soon as SQL finds the first match, it grants the rights listed in that entry and stops the search. All identifiers that do not match a previous entry are compared with the subsequent entry, and if no match occurs, they receive the rights of ("fall through" to) the entry [*,*], if it exists. If no entry has the user identifier [*,*], then unmatched user identifiers are denied all access to the database, table, or column. For this reason, both the entries and their order in the list are important.

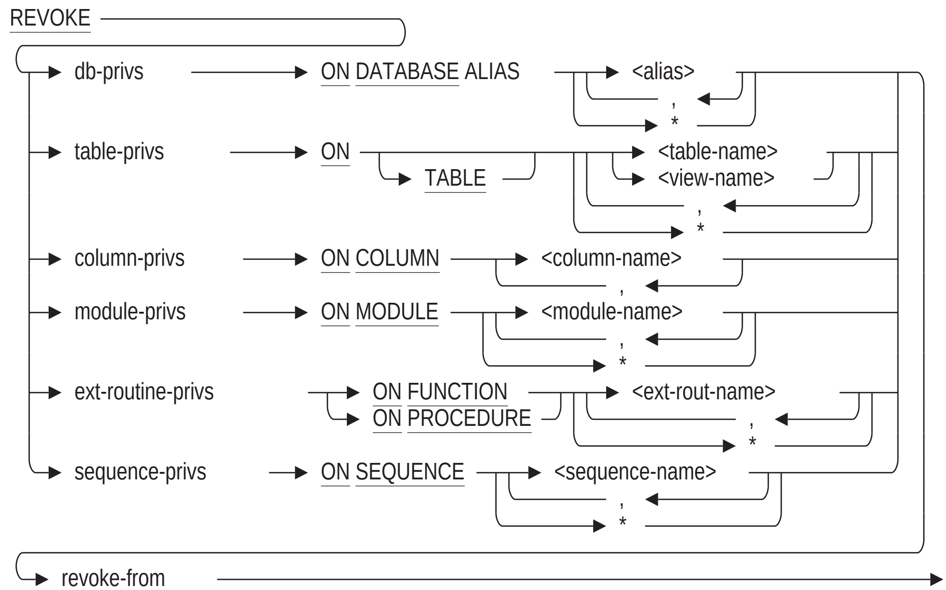
To create or add privileges to an entry to the Oracle Rdb access privilege set for a database, table, view, column, module, external routine, or sequence, see the GRANT Statement in this manual and in the *Oracle Rdb7 SQL Reference Manual*.

Environment

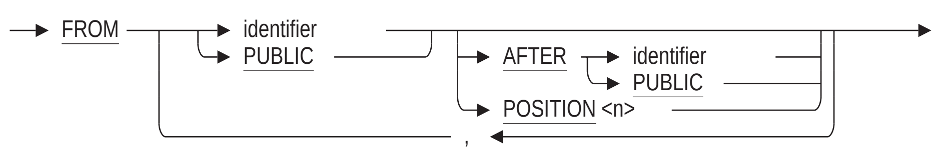
You can use the REVOKE statement:

- In interactive SQL
- Embedded in host language programs to be precompiled
- As part of a nonstored procedure in a nonstored SQL module
- In dynamic SQL as a statement to be dynamically executed

Format



revoke-from =



db-privs=

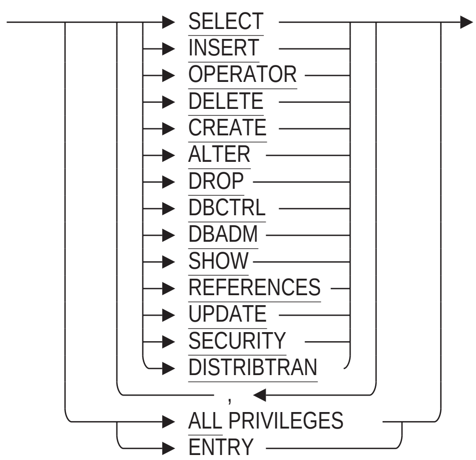
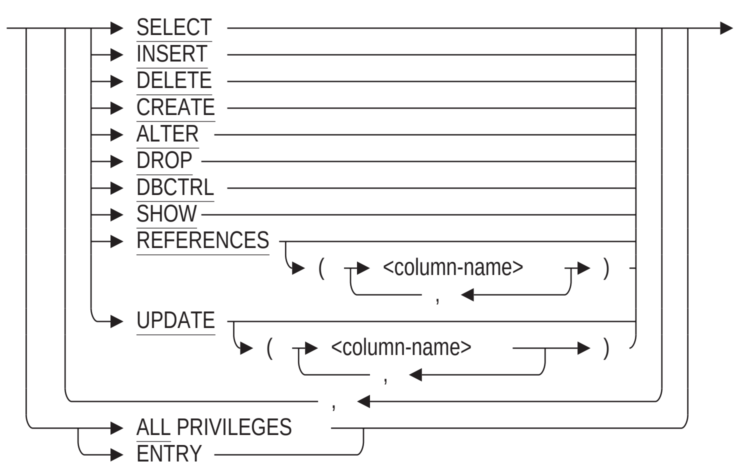
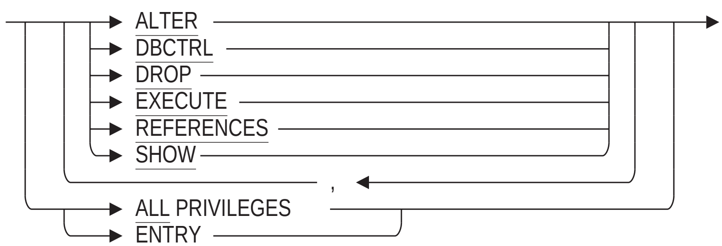


table-privs=

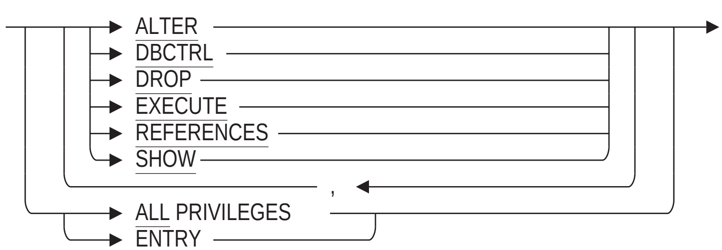


column-privs =

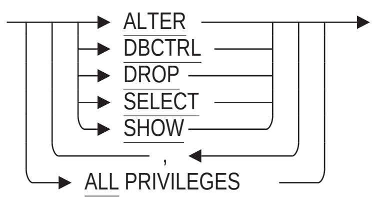
module-privs =



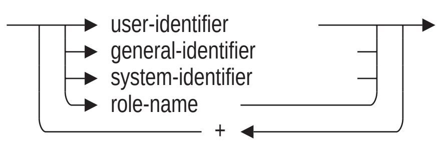
ext-routine-privs =



sequence-privs-ansi =



identifier =



Arguments

db-privs
table-privs
column-privs
module-privs
ext-routine-privs
sequence-privs

Specifies the list of privileges that you want to remove from an existing ACL entry. The operations permitted by a given privilege keyword differ, depending on whether it was granted for a database, table, column, module, external routine, or sequence.

Table 3–5 lists the privilege keywords and their meanings for databases, tables, columns, modules, external routines, and sequences.

For the SELECT, INSERT, and DELETE data manipulation privileges, SQL checks the ACL for the database and for the individual table before allowing access to a specific table. For example, if your SELECT privilege for a database that contains the EMPLOYEES table is revoked, you will not be able to read rows from the table even though you may have SELECT privilege to the EMPLOYEES table itself.

To revoke the data manipulation privileges UPDATE and REFERENCES, you must have at least read access to the database and the appropriate column privilege.

You cannot deny yourself the DBCTRL privilege for a database, table, module, external routine, or sequence that you create.

The SELECT privilege is a prerequisite for all other privileges. If you revoke the SELECT privilege, you effectively deny all privileges, even if they are specified in the privilege list. This restriction may cause REVOKE statements to fail when you might expect them to work. For instance, the following REVOKE statement fails because it tries to revoke the SELECT privilege from the ACL entry for the owner. Because that implicitly denies DBCTRL on the table to the owner, the statement fails.

```
SQL> REVOKE SELECT ON EMPLOYEES FROM serle;
%RDB-E-NO_PRIV, privilege denied by database facility
```

ALL PRIVILEGES

Specifies that SQL should revoke all privileges in the ACL entry. The REVOKE ALL PRIVILEGES statement differs from the REVOKE ENTRY statement in that it does not delete the entire entry from the ACL. The identifier remains, but without any privileges. An empty ACL entry denies all access to users matching the identifier, even if an entry later in the ACL grants PUBLIC access.

ENTRY

Deletes the entire entry in the ACL, including the identifier.

ON SEQUENCE sequence-name

Specifies whether the REVOKE statement applies to ACLs for sequences. You can specify a list of names for any form of the ON clause.

ON DATABASE ALIAS *
ON TABLE *
ON MODULE *
ON FUNCTION *
ON PROCEDURE *
ON SEQUENCE *

Specifies whether the REVOKE statement applies to ACLs for all objects of the specified type. If privileges are denied for the operation on some objects, then the REVOKE is aborted. However, some objects may have protection changes applied.

Usage Notes

No new usage notes.

Examples

Example 1 Revoking DROP Sequence Privileges from a User

```
SQL> CREATE SEQUENCE EMPID;
SQL> SHOW PROTECTION ON SEQUENCE EMPID
Protection on Sequence EMPID
    (IDENTIFIER=[RDB,STUART],ACCESS=SELECT+SHOW+ALTER+DROP+DBCTRL)
    (IDENTIFIER=[*,*],ACCESS=NONE)
SQL> GRANT SELECT ON SEQUENCE EMPID TO PUBLIC;
SQL> SHOW PROTECTION ON SEQUENCE EMPID;
Protection on Sequence EMPID
    (IDENTIFIER=[RDB,STUART],ACCESS=SELECT+SHOW+ALTER+DROP+DBCTRL)
    (IDENTIFIER=[*,*],ACCESS=SELECT)
SQL> REVOKE DROP ON SEQUENCE EMPID FROM STUART;
SQL> SHOW PROTECTION ON SEQUENCE EMPID;
Protection on Sequence EMPID
    (IDENTIFIER=[RDB,STUART],ACCESS=SELECT+SHOW+ALTER+DBCTRL)
    (IDENTIFIER=[*,*],ACCESS=SELECT)
```

REVOKE Statement: ANSI/ISO-Style

Removes privileges from the Oracle Rdb access privilege set granted by a specific user for a database, table, column, module, external routine, or sequence. Each entry in an ANSI/ISO-style access privilege set consists of an identifier and a list of privileges assigned to the identifier.

- Each identifier specifies a user or the PUBLIC keyword.
- The set of privileges specifies what operations that user or user group can perform on the database, table, column, module, external routine, or sequence.

For ANSI/ISO-style databases, the access privilege set is not order-dependent. The user matches the entry in the access privilege set, receives whatever privileges have been granted on the database, table, column, module, external routine, or sequence and receives the privileges defined for PUBLIC. A user without an entry in the access privilege set receives only the privileges defined for PUBLIC. A user loses a privilege when there are no users who grant that privilege to the user. The PUBLIC identifier always has an entry in the access privilege set, even if PUBLIC has no access to the database, table, column, module, external routine, or sequence.

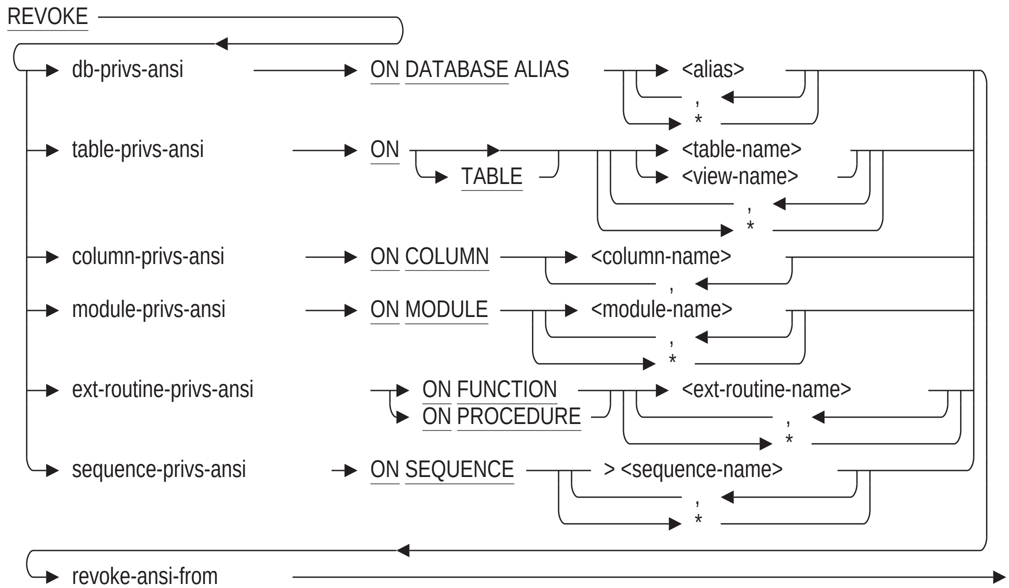
To create or add privileges to an entry to the Oracle Rdb access privilege set for a database, table, view, column, module, external routine, or sequence, see the GRANT Statement: ANSI/ISO-Style.

Environment

You can use the REVOKE statement:

- In interactive SQL
- Embedded in host language programs to be precompiled
- As part of a nonstored procedure in a nonstored SQL module
- In dynamic SQL as a statement to be dynamically executed

FORMAT



db-privs-ansi =

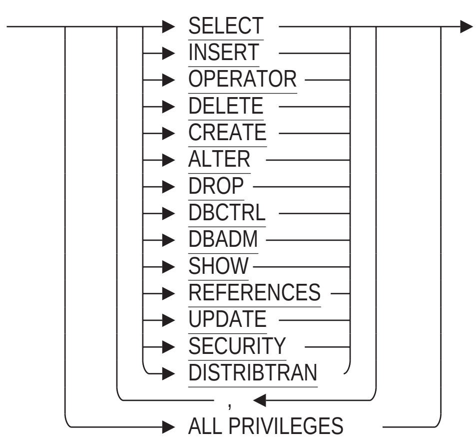
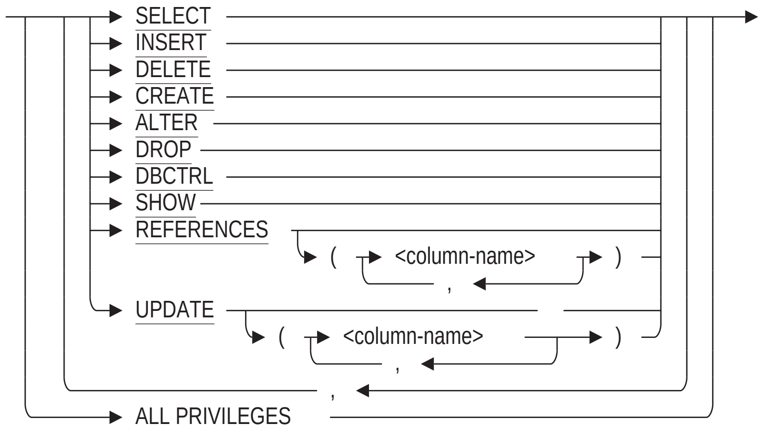
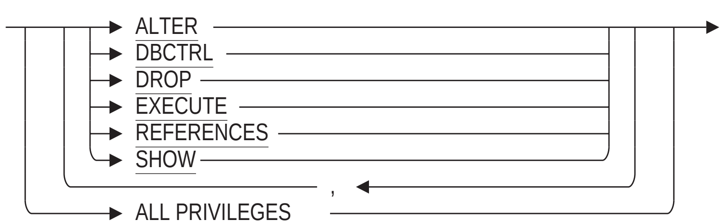


table-privs-ansi =



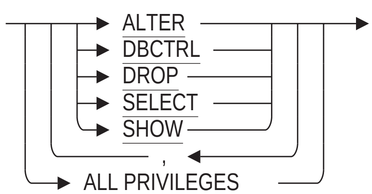
column-privs-ansi =

module-privs-ansi =

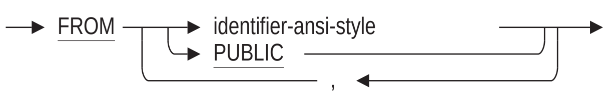


ext-routine-privs-ansi =

sequence-privs-ansi =



revoke-ansi-from =



identifier-ansi-style =

Arguments

db-privs-ansi
table-privs-ansi
column-privs-ansi
module-privs-ansi
ext-routine-privs-ansi
sequence-privs-ansi
Specifies the list of privileges that you want to remove from an existing access privilege set entry. The operations permitted by a given privilege keyword differ, depending on whether it was granted for a database, table, column, module, external routine, or sequence.

Table 3–5 lists the privilege keywords and their meanings for databases, tables, columns, modules, external routines, and sequences.

ON SEQUENCE sequence-name
Specifies whether the REVOKE statement applies to access privilege sets for sequences. You can specify a list of names for any form of the ON clause.

ON DATABASE ALIAS *
ON TABLE *
ON MODULE *
ON FUNCTION *
ON PROCEDURE *
ON SEQUENCE *
Specifies whether the REVOKE statement applies to ACLs for all objects of the specified type. If privileges are denied for the operation on some objects, then the revoke operation is aborted. However, some objects may have protection changes applied.

ALL PRIVILEGES
Specifies that SQL should revoke all privileges in the access privilege set entry.

Usage Notes

No new usage notes.

Examples

Example 1 Revoking DROP Privilege from a Sequence for a User

```
SQL> GRANT ALL PRIVILEGES ON SEQUENCE EMPLOYEE_ID_GEN TO ISTEWART;
```

```
SQL> SHOW PROTECTION ON SEQUENCE EMPLOYEE_ID_GEN;
Protection on Sequence EMPLOYEE_ID_GEN
    (IDENTIFIER=[RDB,ISTEWART],ACCESS=SELECT+SHOW+ALTER+DROP+DBCTRL)
    (IDENTIFIER=[*,*],ACCESS=NONE)
SQL> GRANT SELECT ON SEQUENCE EMPLOYEE_ID_GEN TO PUBLIC;
SQL> SHOW PROTECTION ON SEQUENCE EMPLOYEE_ID_GEN;
Protection on Sequence EMPLOYEE_ID_GEN
    (IDENTIFIER=[RDB,ISTEWART],ACCESS=SELECT+SHOW+ALTER+DROP+DBCTRL)
    (IDENTIFIER=[*,*],ACCESS=SELECT)
SQL> REVOKE DROP ON SEQUENCE EMPLOYEE_ID_GEN FROM ISTEWART;
Protection on Sequence EMPLOYEE_ID_GEN
    (IDENTIFIER=ISTEWART,ACCESS=SELECT+SHOW+ALTER+DBCTRL)
    (IDENTIFIER=PUBLIC,ACCESS=SELECT)
```

REVOKE Statement: Roles

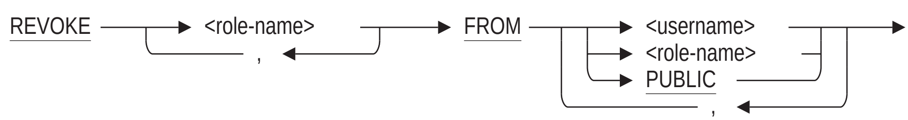
Revokes a role from another user or role and provides internal security checking.

Environment

You can use the REVOKE statement:

- In interactive SQL
- Embedded in host language programs to be precompiled
- As part of a procedure in an SQL module
- In dynamic SQL as a statement to be dynamically executed

Format



Arguments

role-name
The name of an existing role created with the CREATE ROLE statement or that can be created automatically. (If the role name exists as an operating system group or rights identifier, then Oracle Rdb will automatically create the role when you issue the REVOKE statement. A role that is created automatically is identified externally.)

FROM username
FROM role-name
FROM PUBLIC
Specifies the user, role, or the PUBLIC user from which the specified role is to be revoked.

Usage Notes

- You must have the SECURITY privilege on the database to revoke a role from a user or another role.

Examples

Example 1 Granting and Revoking Roles

```
SQL> -- Optionally, create three users and two roles.
SQL> -- Oracle Rdb automatically generates users and
SQL> -- roles if they are identified externally.
SQL> CREATE USER ABLOWNEY IDENTIFIED EXTERNALLY;
SQL> CREATE USER BGREMBO IDENTIFIED EXTERNALLY;
SQL> CREATE USER LWARD IDENTIFIED EXTERNALLY;
SQL> CREATE ROLE SALES_MANAGER IDENTIFIED EXTERNALLY;
SQL> CREATE ROLE DIVISION_MANAGER IDENTIFIED EXTERNALLY;
SQL> -- Grant the SALES_MANAGER role to users ABLOWNEY and
SQL> -- BGREMBO. Also grant the SALES_MANAGER role to the
SQL> -- DIVISION MANAGER ROLE.
SQL> GRANT SALES_MANAGER TO ABLOWNEY, BGREMBO, DIVISION_MANAGER;
SQL> -- Grant the DIVISION_MANAGER role to LWARD. LWARD now
SQL> -- has both the SALES_MANAGER and DIVISION_MANAGER roles.
SQL> GRANT DIVISION_MANAGER TO LWARD;
SQL> -- Revoke the DIVISION_MANAGER role from LWARD. He has
SQL> -- left the company.
SQL> REVOKE DIVISION_MANAGER FROM LWARD;
SQL> -- Grant the DIVISION_MANAGER role to BGREMBO.  She
SQL> -- has been promoted to division manager.
SQL> GRANT DIVISION_MANAGER TO BGREMBO;
```

SET COMPOUND TRANSACTIONS Statement

Allows you to control the SQL behavior for starting a default transaction for a compound statement.

By default, if there is no current transaction, SQL starts a transaction before executing a compound statement or stored procedure. However, this might conflict with the actions within the procedure, or it might start a transaction for no reason if the procedure body does not perform any database access.

Environment

You can use the SET COMPOUND TRANSACTIONS statement:

- In interactive SQL
- In dynamic SQL as a statement to be dynamically executed

Format

```
SET COMPOUND TRANSACTION ──► int-ext-val
```

Argument

int-ext-value
A string literal or host variable containing the keyword 'INTERNAL' or 'EXTERNAL'. These keywords can be in any case (uppercase, lowercase, or mixed case). If the value is EXTERNAL, then SQL starts a transaction before executing the procedure. If the value is set to INTERNAL, then SQL allows the procedure to start a transaction as required by the procedure execution.

By default, SQL starts a transaction before executing a compound statement if there is no current transaction.

Example

Example 1 Enabling and Disabling Transaction Starting

```
SQL> SET COMPOUND TRANSACTIONS 'INTERNAL';
SQL> CALL start_txn_and_commit ();
```

```
SQL> SET COMPOUND TRANSACTIONS 'EXTERNAL';
SQL> CALL update_employees (...);
```

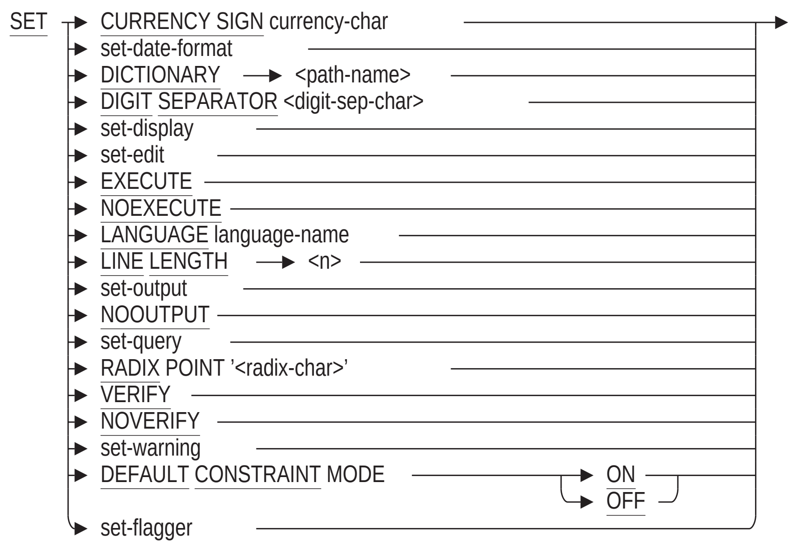
SET DISPLAY Statement

Controls the output of header information.

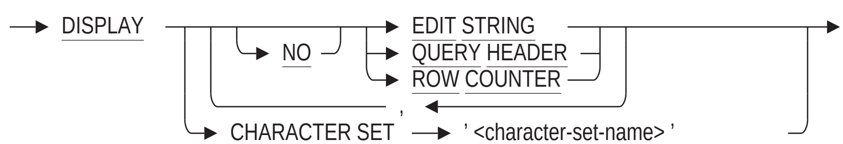
Environment

You can use the SET DISPLAY statement in interactive SQL only.

Format



set-display =



Usage Notes

- The SET DISPLAY statement accepts options to enable or disable parts of the formatted output generated by various statements in interactive SQL:
  - EDIT STRING enables the usage of column edit strings to format values for the SELECT statement. Use NO EDIT STRING to disable the use of the column edit strings.
  - QUERY HEADER enables the printed header generated by the SELECT, CALL, FETCH, and PRINT statements. Use NO QUERY HEADER to disable this header.
  - ROW COUNTER enables the total count reported by SELECT, DELETE, INSERT, and UPDATE statements. Use NO ROW COUNTER to disable the trailing count message.

  The defaults are to use edit strings, display the query header, and report a row count message. More than one option can be specified, separated by commas. However, you cannot specify both the option and its negated form in one statement, as demonstrated in the following example:

  ```
  SQL> SET DISPLAY QUERY HEADER, NO QUERY HEADER
  %SQL-F-MULTSPECATR, Multiple specified attribute.
  "QUERY HEADER" was specified more than once
  ```

Examples

Example 1 Using the SET DISPLAY Statement

The following example shows the effect of the SET DISPLAY statement. It uses the SHOW DISPLAY command to report the current settings.

```
SQL> ATTACH 'FILENAME mf_personnel';
SQL>
SQL> CREATE DOMAIN money INTEGER(2) EDIT STRING '$$$,$$9.99';
SQL> CREATE TABLE temp_emp (id INTEGER, sal money);
SQL>
SQL> SELECT * FROM work_status;
 STATUS_CODE   STATUS_NAME   STATUS_TYPE
 0             INACTIVE      RECORD EXPIRED
 1             ACTIVE        FULL TIME
 2             ACTIVE        PART TIME
3 rows selected
SQL>
SQL> SET DISPLAY NO ROW COUNTER;
```

```
SQL> SHOW DISPLAY
Output of the query header is enabled
Output of the row counter is disabled
Output using edit strings is enabled
HELP page length is set to 24 lines
Line length is set to 132 bytes
SQL>
SQL> SELECT * FROM work_status;
 STATUS_CODE   STATUS_NAME   STATUS_TYPE
 0             INACTIVE      RECORD EXPIRED
 1             ACTIVE        FULL TIME
 2             ACTIVE        PART TIME
SQL> INSERT INTO temp_emp (id) VALUES (0);
SQL> INSERT INTO temp_emp (id, sal)
cont>     SELECT employee_id, MAX(salary_amount)
cont>     FROM salary_history GROUP BY employee_id;
SQL> UPDATE temp_emp SET id = NULL WHERE id <= 0;
SQL> DELETE FROM temp_emp WHERE id IS NULL;
SQL>
SQL> SET DISPLAY ROW COUNTER;
SQL> SHOW DISPLAY
Output of the query header is enabled
Output of the row counter is enabled
Output using edit strings is enabled
HELP page length is set to 24 lines
Line length is set to 132 bytes
SQL>
SQL> SELECT * FROM work_status;
 STATUS_CODE   STATUS_NAME   STATUS_TYPE
 0             INACTIVE      RECORD EXPIRED
 1             ACTIVE        FULL TIME
 2             ACTIVE        PART TIME
3 rows selected
SQL>
SQL> SET DISPLAY NO QUERY HEADER;
SQL> SHOW DISPLAY
Output of the query header is disabled
Output of the row counter is enabled
Output using edit strings is enabled
HELP page length is set to 24 lines
Line length is set to 132 bytes
SQL>
SQL> DECLARE :res INTEGER;
SQL>
SQL> -- This omits the query header for the SELECT statement
```

```
SQL> SELECT * FROM work_status;
 0             INACTIVE      RECORD EXPIRED
 1             ACTIVE        FULL TIME
 2             ACTIVE        PART TIME
3 rows selected
SQL>
SQL> -- This omits the query header for the PRINT statement
SQL> PRINT :res;
           0
SQL> PRINT 'This is a print line';
 This is a print line
SQL>
SQL> CREATE MODULE call_sample
cont>     LANGUAGE SQL
cont>     PROCEDURE add_one (IN :a INTEGER, OUT :b INTEGER);
cont>     SET :b = :a + 1;
cont> END MODULE;
SQL> -- This omits the query header for the OUT/INOUT parameters for CALL
SQL> CALL add_one (100, :res);
         101
SQL>
SQL> DECLARE c CURSOR FOR SELECT * FROM work_status;
SQL> OPEN c;
SQL> -- This omits the query headers for the variables fetched
SQL> FETCH c;
 0             INACTIVE      RECORD EXPIRED
SQL> SET DISPLAY QUERY HEADER;
SQL> SHOW DISPLAY
Output of the query header is enabled
Output of the row counter is enabled
Output using edit strings is enabled
HELP page length is set to 24 lines
Line length is set to 132 bytes
SQL> -- This outputs the query headers for the variables fetched
SQL> FETCH c;
 STATUS_CODE   STATUS_NAME   STATUS_TYPE
 1             ACTIVE        FULL TIME
SQL> CLOSE c;
SQL>
SQL> TRUNCATE TABLE temp_emp;
SQL> INSERT INTO temp_emp (id, sal)
cont>     SELECT employee_id, AVG(salary_amount)
cont> FROM salary_history
cont> WHERE salary_end IS NULL
cont> GROUP BY employee_id;
```

```
100 rows inserted
SQL>
SQL> SELECT * FROM temp_emp ORDER BY id LIMIT TO 3 ROWS;
          ID          SAL
         164   $51,712.00
         165   $11,676.00
         166   $18,497.00
3 rows selected
SQL>
SQL> SET DISPLAY NO EDIT STRING;
SQL> SHOW DISPLAY
Output of the query header is enabled
Output of the row counter is enabled
Output using edit strings is disabled
HELP page length is set to 24 lines
Line length is set to 132 bytes
SQL>
SQL> SELECT * FROM temp_emp ORDER BY id LIMIT TO 3 ROWS;
          ID          SAL
         164     51712.00
         165     11676.00
         166     18497.00
3 rows selected
SQL>
SQL> SET DISPLAY EDIT STRING;
SQL> SHOW DISPLAY
Output of the query header is enabled
Output of the row counter is enabled
Output using edit strings is enabled
HELP page length is set to 24 lines
Line length is set to 132 bytes
SQL>
SQL> SELECT * FROM temp_emp ORDER BY id LIMIT TO 3 ROWS;
          ID          SAL
         164   $51,712.00
         165   $11,676.00
         166   $18,497.00
3 rows selected
```

Note: The SHOW DISPLAY statement may also report the current line length (which can be changed using the SET LINE LENGTH command) and the HELP page length (which is automatically established for the interactive HELP command).

SET FLAGS Statement

Allows enabling and disabling of database system debug flags for the current session.

The literal or host variable passed to this command can contain a list of keywords, or negated keywords, separated by commas. Spaces are ignored. The keywords may be abbreviated to an unambiguous length.

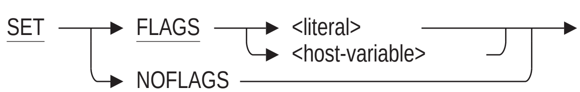
Note: Oracle Corporation reserves the right to add new keywords to the SET FLAGS statement in any release or update to Oracle Rdb, which may change this unambiguous length. Therefore, it is recommended that the full keyword be used in applications.

Environment

You can use the SET FLAGS statement:

- In interactive SQL
- In dynamic SQL as a statement to be dynamically executed

Format



Arguments

FLAGS

Specifies whether or not a database system debug flag is set. The flags are shown in Table 3–6. Unless otherwise indicated in the table, the Debug Flags Equivalent sets the RDMS$DEBUG_FLAGS logical name to the behavior listed under the keyword.

In addition, the keywords (and negated keywords) listed in the table can be specified as the equivalence string for the RDMS$SET_FLAGS logical name. See Section 2.2.4 for details.

To set the query mode with a logical name, define the RDMS$BIND_OUTLINE_MODE logical name to the desired mode number.

Table 3–6 Debug Flag Keywords

KEYWORD	Negated Keyword	Debug Flags Equivalent[1]	Comment
AUTO_OVERRIDE	NOAUTO_OVERRIDE	None	Allows a user with the DBADM (administrator) privilege to insert or update a column defined as AUTOMATIC. See Example 9.
BLR	NOBLR	Bcn[2]	Displays the BLR request for the query.
BLR_NOFORMAT	NOBLR_NOFORMAT	Bn[2]	Inhibits offset numbering and other formatting of the BLR display when used with the BLR keyword.
CARDINALITY	NOCARDINALITY	K[2]	Displays cardinality updates.
CONTROL_BITS	NOCONTROL_BITS	Bc[2]	Displays a decoding of the BLR$K_CONTROL_BITS semantic flags when used with the BLR keyword.
COSTING	NOCOSTING	Oc[2]	Displays traces on optimizer costing.
CHRONO_FLAG	NOCHRONO_FLAG	Xc[2]	Forces timestamp-before-dump display.
CURSOR_STATS	NOCURSOR_STATS	Og[2]	Displays general cursor statistics for the optimizer.
DATABASE_PARAMETERS	NODATABASE_PARAMETERS	P[2]	Displays the database parameter buffer during ATTACH, CREATE, ALTER, and DISCONNECT statements.
ESTIMATES	NOESTIMATES	O[2]	Displays the optimizer estimates.

Table 3–6 (Cont.) Debug Flag Keywords

KEYWORD	Negated Keyword	Debug Flags Equivalent[1]	Comment
EXECUTION	NOEXECUTION	E[2]	Displays an execution trace from the dynamic optimizer. For a sequential retrieval from a table that is strictly partitioned, this includes a count and a list of the selected partitions each time the query executes. The EXECUTION keyword can be followed by a numeric value in parentheses. This represents the number of lines to display for stopping the execution trace for the query execution. There can be no spaces between the keyword and the parameter. The default is 100.
IGNORE_OUTLINE	NOIGNORE_OUTLINE	None	Ignores outlines defined in the database. The IGNORE_OUTLINE keyword has the same action as setting the RDMS$BIND_OUTLINE_FLAGS logical name to 1.
INDEX_COLUMN_GROUP	NOINDEX_COLUMN_GROUP	None	Enables leading index columns as workload column groups. This may increase solution cardinality accuracy. See the Usage Notes for details.
INDEX_STATS	NOINDEX_STATS	Ai[2]	Enables debug flags output for the progress of an ALTER, CREATE, or DROP INDEX statement.
INTERNALS	NOINTERNALS	I[2]	Enables debug flags output for internal queries such as constraints and triggers. It can be used in conjunction with other keywords such as STRATEGY, BLR, and EXECUTION.
ITEM_LIST	NOITEM_LIST	H[2]	Displays item list information passed in for the database queries and as compile-time query options.
MAX_STABILITY	NOMAX_STABILITY	none	Enables maximum stability; the dynamic optimizer is not allowed. The MAX_STABILITY keyword has the same action as the RDMS$MAX_STABILITY logical name.

Table 3–6 (Cont.) Debug Flag Keywords

KEYWORD	Negated Keyword	Debug Flags Equivalent[1]	Comment
MBLR	NOMBLR	M[2]	Displays the MBLR request for each DDL statement.
MODE(n)	NOMODE	See Usage Notes	Allows you to specify which query outline should be used by specifying the mode value of that query outline. The value of n can be any positive or negative integer, or n can be omitted. If you specify MODE but omit n, the default is MODE(1). If you specify MODE(0) or NOMODE, it disables the display of the mode in the SHOW FLAGS statement output. MODE(0) is the default for Oracle Rdb generated outlines.
OLD_COST_ MODEL	NOOLD_COST_ MODEL	none	Enables the old cost model. The OLD_ COST_MODEL keyword has the same action as the RDMS$USE_OLD_COST_ MODEL logical name.
OUTLINE	NOOUTLINE	Ss[2]	Displays the query outline for this query.
PREFIX[3]	NOPREFIX	Bn[4]	Used with the BLR keyword to inhibit offset numbering and other formatting of binary language representation display.
REQUEST_NAMES	NOREQUEST_ NAMES	Sn[2]	Displays the names of user requests, triggers, and constraints.
REVERSE_SCAN[3]	NOREVERSE_SCAN	None	Enables the reverse index scan strategy. The NOREVERSE_SCAN keyword has the same action as the RDMS$DISABLE_REVERSE_SCAN logical name.
SCROLL_ EMULATION	NOSCROLL_ EMULATION	L[2]	Disables scrolling for the old style LIST OF BYTE VARYING (segmented string) format, which is an RDMS$DIAG_ FLAGS option.

Table 3–6 (Cont.) Debug Flag Keywords

KEYWORD	Negated Keyword	Debug Flags Equivalent[1]	Comment
SEQ_CACHE(n)	NOSEQ_CACHE	None	Adjusts the sequence cache size for the process issuing the SET FLAGS statement. The value n must be a numeric value greater than 2. (Specifying a value of 1 is equivalent to specifying NOSEQ_CACHE.) Use SEQ_CACHE to override the CACHE setting for all sequences subsequently referenced by the application. The new cache size does not affect any sequence that has already been referenced, or any sequence defined as NOCACHE.
SOLUTIONS	NOSOLUTIONS	Os[2]	Displays traces on optimizer solutions.
SORTKEY_EXT	NOSORTKEY_EXT	None	Reports if the ORDER BY (or SORTED BY) clause refers to external (constant) values only. The SORTKEY_EXT flag has the same action as setting the RDMS$DIAG_FLAGS logical name to S.
SORT_STATISTICS	NOSORT_STATISTICS	R[2]	Displays sort statistics during execution.
STOMAP_STATS	NOSTOMAP_STATS	As[2]	Displays the processing of storage maps for any tables that refer to the dropped storage area. The output is prefixed with "~As". This has the same effect as setting the RDMS$DEBUG_FLAGS logical name to As.
STRATEGY	NOSTRATEGY	S[2]	Shows the optimizer strategy. If a table is strictly partitioned, the text "(partitioned scan#nn)" appears after the table name, where nn indicates the leaf number for a sequential scan (there may be several within a single query).
TRACE	NOTRACE	Xt[2]	Enables output from the SQL TRACE statement.
TRANSACTION_PARAMETERS	NOTRANSACTION_PARAMETERS	T[2]	Displays the transaction parameter buffer during SET TRANSACTION, COMMIT, and ROLLBACK statements, and during stored procedure compilation.

Table 3–6 (Cont.) Debug Flag Keywords

KEYWORD	Negated Keyword	Debug Flags Equivalent[1]	Comment
TRANSITIVITY	NOTRANSITIVITY	None	Enables transitivity between selections and join predicates. The NOTRANSITIVITY keyword has the same action as the RDMS$DISABLE_TRANSITIVITY logical name.
VALIDATE_ ROUTINE	NOVALIDATE_ ROUTINE	None	Enables revalidation of an invalidated stored procedure or stored function. The VALIDATE_ROUTINE keyword has the same action as the RDMS$VALIDATE_ROUTINE logical name.
WARN_INVALID	NOWARN_INVALID	Xw[2]	Reports invalidated objects during the ALTER INDEX, DROP INDEX, DROP TABLE, and DROP MODULE statements.
ZIGZAG_MATCH[3]	NOZIGZAG_ MATCH	None	Enables zigzag key skip on both outer and inner match loops. When you specify the ZIGZAG_MATCH keyword with the NOZIGZAG_OUTER keyword, it disables zigzag key skip on the outer loop (and has the same action as setting the RDMS$DISABLE_ZIGZAG_MATCH logical name to 1). The NOZIGZAG_MATCH keyword disables zigzag key skip on both outer and inner match loops (and has the same action as setting the RDMS$DISABLE_ZIGZAG_MATCH logical name to 2).

Table 3–6 (Cont.) Debug Flag Keywords

KEYWORD	Negated Keyword	Debug Flags Equivalent[1]	Comment
ZIGZAG_OUTER[3]	NOZIGZAG_OUTER	None	Enables zigzag key skip on the outer loop. See the entry for ZIGZAG_MATCH for information on the action taken when you specify ZIGZAG_OUTER and ZIGZAG_MATCH together.

[1] RDMS$DEBUG_FLAGS logical name.

[2] Equivalent to the keyword listed in the Keyword column.

[3] Enabled by default.

[4] Equivalent to the keyword listed in the Negated Keyword column.

Usage Notes

- The SET NOFLAGS statement disables all currently enabled flags. It is equivalent to SET FLAGS 'NONE'.
- The CRONO_FLAG keyword has been replaced with CHRONO_FLAG. However, the CRONO_FLAG keyword is still valid.
- To set the AUTO_OVERRIDE keyword, you must have the DBADM (administrator) privilege on the database. The DBADM privilege can be granted explicitly or can be inherited from the SYSTEM privilege.

  If you do not have the required privilege, then the SET FLAG statement fails and returns the NO_PRIV error.
- The AUTO_OVERRIDE keyword is most often required when executing the AUTOMATIC value expression has resulted in incorrect data being stored in the table. See Example 9 in the Examples section.
- There is no debug flags equivalent for the MODE(n) or NOMODE keywords. Instead, you can use the RDMS$BIND_OUTLINE_MODE logical name.
- When a generated outline is added to the database it will only be used when the mode is set, either by the SET FLAGS statement or by using the logical name RDMS$BIND_OUTLINE_MODE.
- Routines, query outlines, and triggers can become invalid due to the following events:
  - When a table is dropped using the CASCADE option, any procedure or function that references the table is marked invalid.

- When a table is dropped (using either the CASCADE or RESTRICT options) any query outline that references the table is marked as invalid.
- When a module is dropped using the CASCADE option, any procedure, function, or query outline that references the module is marked invalid. A query outline references a module when it uses a temporary table declared at the module level.
- When an index is dropped, or altered to have MAINTENANCE IS DISABLED, any query outline that references the index is marked as invalid.

- When you use the INDEX_COLUMN_GROUP keyword, applications can make better use of the index column group information specified in indexes. When you do not use this keyword, the Oracle Rdb optimizer always seems to estimate much higher cardinalities for the chosen solution if the selection predicate specifies only some of the leading segments on a multisegment index. This happens, for instance, if you specify an equality on the first segment of a two-segment index.

  This slight overestimation is not a significant problem on relatively small tables but becomes a more significant problem when the select operation involves a sort (in particular, the OpenVMS SORT facility) where the sort buffer is preallocated based on its estimated cardinality of the solution. See Example 8 in the Examples section.

- You might use the SEQ_CACHE keyword when you are loading many rows with the RMU Load command. This command is most efficient when all of the sequence values are allocated in large batches. For example:

```
$ DEFINE RDMS$SET_FLAGS   "SEQ_CACHE(10000)"
$ RMU/LOAD/COMMIT_EVERY=50000 DATABASE TABLE FILE
```

  In this example, it is assumed that an AUTOMATIC column is defined such that SEQUENCE.NEXTVAL is executed.

Examples

Example 1 Using the MODE(n) Flag

```
SQL> SET FLAGS 'MODE(10), OUTLINE';
SQL> SHOW FLAGS
Alias RDB$DBHANDLE:
Flags currently set for Oracle Rdb:
   PREFIX,OUTLINE,MODE(10)
```

```
SQL> SELECT COUNT(*) FROM EMPLOYEES;
-- Rdb Generated Outline : 30-MAY-1997 16:35
create outline QO_B3F54F772CC05435_0000000A
id 'B3F54F772CC054350B2B454D95537995'
mode 10
as (
  query (
-- For loop
    subquery (
      subquery (
        EMPLOYEES 0     access path index       EMP_EMPLOYEE_ID
        )
      )
    )
  )
compliance optional     ;
         100
1 row selected
```

Example 2 Using the WARN_INVALID Debug Flag

```
SQL> SET FLAGS 'WARN_INVALID';
SQL> SHOW FLAGS;
Alias RDB$DBHANDLE:
Flags currently set for Oracle Rdb:
   PREFIX,WARN_INVALID
SQL> -- warning because of dependencies
SQL> DROP TABLE T1 CASCADE;
~Xw: Routine "P3" marked invalid
~Xw: Routine "P2" marked invalid
~Xw: Routine "P1" marked invalid
SQL>
SQL> -- Create an outline that references an INDEX.
SQL> CREATE TABLE T1 (A INTEGER, B INTEGER);
SQL> CREATE INDEX I1 ON T1 (A);
SQL> CREATE OUTLINE QO1
cont> ID '19412AB61A7FE1FA6053F43F8F01EE6D'
cont> MODE 0
cont> AS (
cont>   QUERY (
cont>     SUBQUERY (
cont>       T1 0    ACCESS PATH INDEX       I1
cont>       )
cont>     )
cont>   )
```

```
cont> COMPLIANCE OPTIONAL;
SQL>
SQL> -- Warning because of disabled index
SQL> ALTER INDEX I1
cont>     MAINTENANCE IS DISABLED;
~Xw: Outline "QO1" marked invalid (index "I1" disabled)
SQL> SHOW OUTLINE QO1;
     QO1
    Object has been marked INVALID
 Source:
CREATE OUTLINE QO1
ID '19412AB61A7FE1FA6053F43F8F01EE6D'
MODE 0
AS (
  QUERY (
    SUBQUERY (
      T1 0    ACCESS PATH INDEX       I1
      )
    )
  )
COMPLIANCE OPTIONAL;
```

Example 3 Specifying Multiple Flags

```
SQL> SET FLAGS 'STRATEGY, EXECUTION';
SQL> SHOW FLAGS;
Alias RDB$DBHANDLE:
Flags currently set for Oracle Rdb:
   STRATEGY,PREFIX,EXECUTION(100)
```

Example 4 Using a Host Variable to Set Flags

The following example demonstrates the use of host variables in interactive SQL, as well as literal strings with multiple options to enable and disable flags. The same SET FLAGS commands can also be used in dynamic SQL.

```
SQL> SHOW FLAGS;
Alias RDB$DBHANDLE:
Flags currently set for Oracle Rdb:
   PREFIX
SQL> -- Declare a host variable to be used with SET FLAGS
SQL> DECLARE :hv char(40);
SQL> -- Assign a value to the variable
SQL> BEGIN
cont> SET :hv = 'strategy, outline';
```

```
cont> END;
SQL> -- Use the host variable to enable or disable flags
SQL> SET FLAGS :hv;
SQL> SHOW FLAGS
Alias RDB$DBHANDLE:
Flags currently set for Oracle Rdb:
   STRATEGY,PREFIX,OUTLINE
SQL> -- use a string literal directly with the SET FLAGS statement
SQL> set flags ‘noprefix,execution(10)’;
SQL> show flags

Alias RDB$DBHANDLE:
Flags currently set for Oracle Rdb:
  STRATEGY,OUTLINE,EXECUTION(10)
```

Example 5 Using the INTERNAL Keyword to Display Trigger Actions

```
SQL> -- The following code shows the strategy used by the trigger
SQL> -- actions on the AFTER DELETE trigger on EMPLOYEES
SQL> SET FLAGS 'STRATEGY, INTERNALS, REQUEST_NAMES';
SQL> SHOW FLAGS
Alias RDB$DBHANDLE:
Flags currently set for Oracle Rdb:
   INTERNALS,STRATEGY,PREFIX,REQUEST_NAMES
SQL> DELETE FROM EMPLOYEES WHERE EMPLOYEE_ID = '00164';
~S: Trigger name  EMPLOYEE_ID_CASCADE_DELETE
Get     Temporary relation      Retrieval by index of relation DEGREES
  Index name  DEG_EMP_ID [1:1]
~S: Trigger name  EMPLOYEE_ID_CASCADE_DELETE
Get     Temporary relation      Retrieval by index of relation JOB_HISTORY
  Index name  JOB_HISTORY_HASH [1:1]
~S: Trigger name  EMPLOYEE_ID_CASCADE_DELETE
Get     Temporary relation      Retrieval by index of relation SALARY_HISTORY
  Index name  SH_EMPLOYEE_ID [1:1]
~S: Trigger name  EMPLOYEE_ID_CASCADE_DELETE
Conjunct        Get     Retrieval by index of relation DEPARTMENTS
  Index name  DEPARTMENTS_INDEX [0:0]
Temporary relation      Get     Retrieval by index of relation EMPLOYEES
  Index name  EMPLOYEES_HASH [1:1]       Direct lookup
1 row deleted
```

Example 6 Revalidating a Stored Procedure

This example shows the revalidation of a stored procedure. When the stored routine is successfully prepared (but not executed), the setting of VALIDATE_ROUTINE causes the entry for this routine in the RDB$ROUTINES system relation to set a valid value.

```
SQL> SET TRANSACTION READ WRITE;
SQL> SET FLAGS 'VALIDATE_ROUTINE';
SQL> SET NOEXECUTE;
SQL> CALL ADD_EMPLOYEE ('Smith');
SQL> SET EXECUTE
SQL> COMMIT;
```

In this example, the use of SET NOEXECUTE in interactive SQL allows the stored routine to be successfully compiled, but it is not executed.

Example 7 Specifying an EXECUTION Keyword

```
SQL> SET FLAGS 'EXECUTION(1000)';
SQL> SHOW FLAGS;
Alias RDB$DBHANDLE:
Flags currently set for Oracle Rdb:
   PREFIX,EXECUTION(1000)
```

Example 8 Using the INDEX_COLUMN_GROUP Keyword

```
SQL> -- The table STUDENTS has an index on the two columns
SQL> -- STU_NUM and COURSE_NUM. When the INDEX_COLUMN_GROUP
SQL> -- keyword is not set, the optimizer uses a fixed
SQL> -- proportion of the table cardinality based on the equality
SQL> -- with the STU_NUM column. In this example, 5134 rows are expected,
SQL> -- when in reality, only 9 are returned by the query.
SQL> CREATE INDEX STUDENT_NDX ON STUDENTS (STU_NUM,COURSE_NUM DESC);
SQL> --
SQL> SELECT STU_NUM FROM STUDENTS
cont>  WHERE STU_NUM = 191270771
cont>  ORDER BY OTHER_COLUMN;
Solutions tried 2
Solutions blocks created 1
Created solutions pruned 0
Cost of the chosen solution   4.5644922E+03
Cardinality of chosen solution   5.1342500E+03
~O: Physical statistics used
Sort
SortId# 7., # Keys 2
  Item# 1, Dtype: 2, Order: 0, Off: 0, Len: 1
  Item# 2, Dtype: 35, Order: 0, Off: 1, Len: 8
  LRL: 32, NoDups:0, Blks:327, EqlKey:0, WkFls: 2
Leaf#01 BgrOnly STUDENTS Card=164296
```

```
  BgrNdx1 STUDENT_NDX [1:1] Fan=14
   191270771
   191270771
   191270771
   191270771
   191270771
   191270771
   191270771
   191270771
SORT(9) SortId# 7, --------------------- Version: V5-000
  Records Input: 9      Sorted: 9       Output: 0
  LogRecLen Input: 32     Intern: 32      Output: 32
  Nodes in SoTree: 5234   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: 4606464 busy
   191270771
9 rows selected
SQL> --
SQL> -- When you use the SET FLAGS statement to set the
SQL> -- INDEX_COLUMN_GROUP keyword, it activates the optimizer
SQL> -- to consider the index segment columns as a workload column
SQL> -- group, compute the statistics for duplicity factor and null
SQL> -- factor dynamically, and then apply them in estimating the
SQL> -- cardinality of the solution.
SQL> --
SQL> SET FLAGS 'INDEX_COLUMN_GROUP';
SQL> -- The following is the optimizer cost estimate and sort output trace
SQL> -- for the previous query with INDEX_COLUMN_GROUP enabled. The optimizer
SQL> -- now estimates a lower cardinality of about 8 rows.
Solutions tried 2
Solutions blocks created 1
Created solutions pruned 0
Cost of the chosen solution   3.8118614E+01
Cardinality of chosen solution   8.3961573E+00
~O: Workload and Physical statistics used
Sort
SortId# 2., # Keys 2
  Item# 1, Dtype: 2, Order: 0, Off: 0, Len: 1
  Item# 2, Dtype: 35, Order: 0, Off: 1, Len: 8
  LRL: 32, NoDups:0, Blks:7, EqlKey:0, WkFls: 2
Leaf#01 BgrOnly STUDENTS Card=164296
  BgrNdx1 STUDENT_NDX [1:1] Fan=14
```

```
   191270771
   191270771
   191270771
   191270771
   191270771
   191270771
   191270771
   191270771
SORT(2) SortId# 2, --------------------- Version: V5-000
  Records Input: 9      Sorted: 9       Output: 0
LogRecLen Input: 32     Intern: 32      Output: 32
Nodes in SoTree: 114    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: 87552 idle
   191270771
9 rows selected
```

Example 9 Using the AUTO_OVERRIDE Keyword

```
SQL> -- Suppose that after year 2000 testing was performed on a
SQL> -- production system, the system date and time were not reset
SQL> -- to the correct date.  This was not noticed until
SQL> -- after transactions for a full day had been stored. To
SQL> -- correct this problem, the database administrator overrides
SQL> -- the READ ONLY characteristic of the AUTOMATIC column and
SQL> -- adjusts the date and time.
SQL> SELECT * FROM ACCOUNTS
cont> WHERE LAST_UPDATE > DATE’2001-1-1’;
       ACCOUNT_NO     LAST_NAME          LAST_UPDATE    CURRENT_BALANCE
       NULL           Smith              200-06-02      100000.000
    1 row selected
SQL> -- Attempts to fix the date and time fail because the
SQL> -- column is AUTOMATIC.
SQL> UPDATE ACCOUNTS
cont>     SET LAST_UPDATE = LAST_UPDATE - INTERVAL'1' YEAR
cont>     WHERE LAST_UPDATE > DATE'2000-1-1';
%RDB-E-READ_ONLY_FIELD, attempt to update the read-only field LAST_UPDATE
SQL> --
SQL> SET FLAGS 'AUTO_OVERRIDE';
SQL> SHOW FLAGS
Alias RDB$DBHANDLE:
```

```
Flags currently set for Oracle Rdb:
   PREFIX,AUTO_OVERRIDE
SQL>--
SQL> -- Fix the date and time.
SQL> UPDATE ACCOUNTS
cont>     SET LAST_UPDATE = LAST_UPDATE - INTERVAL'1' YEAR
cont>     WHERE LAST_UPDATE > DATE'2000-1-1';
1 row updated
SQL>
SQL> SELECT * FROM ACCOUNTS;
           ACCOUNT_NO   LAST_NAME              LAST_UPDATE   CURRENT_BALANCE
                 NULL   Smith                  1999-06-02         100000.000
1 row selected
SQL>
SQL> SET FLAGS 'NOAUTO_OVERRIDE';
```

SET QUIET COMMIT Statement

Allows you to control the error reporting behavior when a COMMIT or ROLLBACK statement is executed although there is no active transaction. By default, if there is no active transaction, SQL raises an error when a COMMIT or ROLLBACK statement is executed. If the SET QUIET COMMIT statement is set to ON, then a COMMIT or ROLLBACK statement executes successfully even when there is no active transaction.

Environment

You can use the SET QUIET COMMIT statement:

- In interactive SQL
- In dynamic SQL as a statement to be dynamically executed

Format

```
SET QUIET COMMIT ──► on-or-off-value
```

Arguments

on-or-off-value

Specifies a string literal or host variable containing the keyword ON or OFF.

The ’ON’ argument specifies that if a COMMIT or ROLLBACK transaction is executed when there is no active transaction, then SQL will not raise an error. The ’OFF’ argument specifies that if a COMMIT or ROLLBACK statement is executed when there is no active transaction, then SQL will raise an error. You can specify the ’ON’ and ’OFF’ arguments using any case (uppercase, lowercase, or mixed case).

By default, if there is no active transaction, SQL raises an error when the COMMIT or ROLLBACK statement is executed. This default is retained for backward compatibility for applications that want to detect this situation.

Usage Notes

- The following options and qualifiers have the same effect as the SET QUIET COMMIT statement in their respective interfaces:

- QUIET COMMIT for the SQL module language header option
- /QUIET_COMMIT and /NOQUIET_COMMIT qualifiers for the SQL module language qualifier
- /SQLOPTIONS=QUIET_COMMIT and /SQLOPTIONS=NOQUIET_COMMIT qualifiers for the SQL language precompiler

- If you issue a COMMIT or ROLLBACK statement within a compound statement, stored procedure, or function, no exception is ever raised when a transaction is not active and you have not issued the SET QUIET COMMIT statement. In effect, the behavior of the SET QUIET COMMIT statement is always active for compound statements, stored procedures, and functions.

Examples

Example 1 Setting the QUIET COMMIT Option On and Off

```
SQL> COMMIT;
%SQL-F-NO_TXNOUT, No transaction outstanding
SQL> SET QUIET COMMIT 'ON';
SQL> ROLLBACK;
SQL> SET QUIET COMMIT 'OFF';
SQL> ROLLBACK;
%SQL-F-NO_TXNOUT, No transaction outstanding
```

SET SESSION AUTHORIZATION Statement

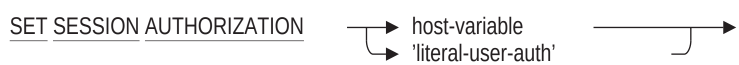
Allows you to transfer the current database attach to another user.

Environment

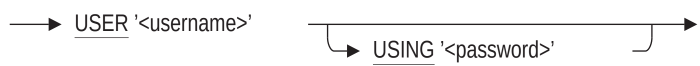
You can use the SET SESSION AUTHORIZATION statement:

- In interactive SQL
- Embedded in host language programs to be precompiled
- As part of a procedure in an SQL module
- In dynamic SQL as a statement to be dynamically executed

Format



literal-user-auth =



Arguments

host-variable
'literal-user-auth'
Specifies the name of the user and the password to whom the database attach is being transferred as a string literal or a host variable. If a host-variable is specified, it must contain the literal-user-auth as a string literal.

USER 'username'
A character string literal that specifies the operating system user name that the database system uses for privilege checking.

USING ’password’

A character string literal that specifies the user’s password for the user name specified in the USER clause.

Usage Notes

- You must have the SELECT privilege on the database to set session authorization.

Examples

Example 1 Reusing the Current Database Attach for Another User

```
SQL> ATTACH ’FILENAME db$:personnel’;
SQL> SET SESSION AUTHORIZATION ’USER ’’SMITHI’’ USING ’’SECRET1’’’;
SQL> SHOW PRIV ON DATABASE RDB$DBHANDLE
Privileges on Alias RDB$DBHANDLE
    (IDENTIFIER=[RDB,SMITHI],ACCESS=SELECT+INSERT+UPDATE+DELETE+SHOW+CREATE+
      ALTER+DROP+DBCTRL+OPERATOR+DBADM+REFERENCES+SECURITY+DISTRIBTRAN)
SQL> SET SESSION AUTHORIZATION ’USER ’’JAIN’’ USING ’’SECRET2’’’;
SQL> SHOW PRIV ON DATABASE RDB$DBHANDLE
Privileges on Alias RDB$DBHANDLE
   (IDENTIFIER=[RDB,JAIN],ACCESS=SELECT+INSERT+UPDATE+DELETE+SHOW+CREATE+ALTER+
      DROP+DBCTRL+OPERATOR+DBADM+REFERENCES+SECURITY+DISTRIBTRAN)
```

SET TRANSACTION Statement

Starts a transaction and specifies its characteristics. A **transaction** is a group of statements whose changes can be made permanent or undone only as a unit. The characteristics specified in a SET TRANSACTION statement affect all transactions until the transaction ends.

A transaction ends with a COMMIT or ROLLBACK statement. If you end the transaction with the COMMIT statement, all changes made to the database by the statements are made permanent. If you end the transaction with the ROLLBACK statement, the statements do not take effect.

You must end the transaction with a COMMIT or ROLLBACK statement before starting or declaring another transaction. If you try to start or declare a transaction while another one is active, SQL generates an error message.

Besides the SET TRANSACTION statement, you can specify the characteristics of a transaction in one of two other ways:

- If you specify the DECLARE TRANSACTION statement, the declarations in the statement take effect when SQL starts a new transaction that is not started by the SET TRANSACTION statement. SQL starts a new transaction with the first executable data manipulation or data definition statement following the DECLARE TRANSACTION, COMMIT, or ROLLBACK statement.
- If you omit both the DECLARE and SET TRANSACTION statements, SQL automatically starts a transaction (using the read / write option) with the first executable data manipulation or data definition statement following a COMMIT or ROLLBACK statement. Thus, you can retrieve and update data without declaring or setting a transaction explicitly.

See the *Oracle Rdb7 SQL Reference Manual* for examples of when you would want to use the DECLARE TRANSACTION statement instead of the SET TRANSACTION statement.

You can specify many options with the SET TRANSACTION statement, including:

- Transaction mode (READ ONLY / READ WRITE)
- Lock specification clause (RESERVING options)
- Horizontal partition specification (RESERVING options)
- Wait mode (WAIT / NOWAIT)

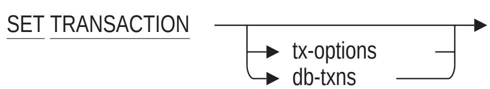
- Isolation level
- Constraint evaluation specification clause
- Multiple sets of all the preceding options for each database involved in the transaction (ON ... AND ON)

Environment

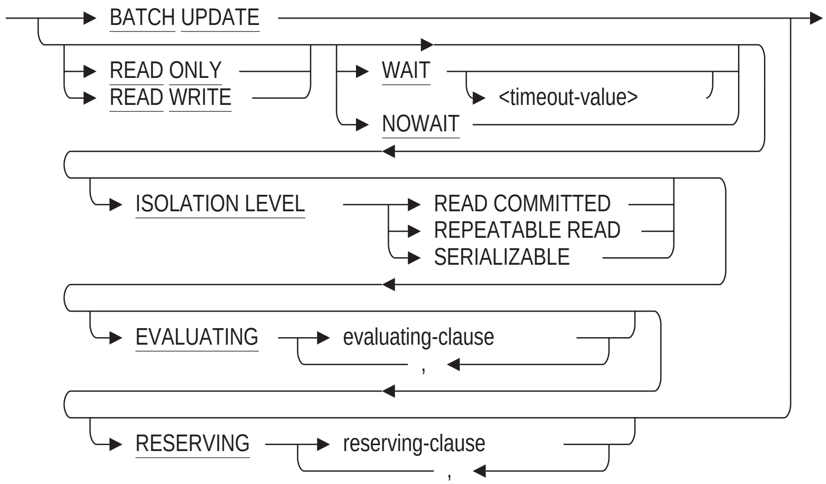
You can use the SET TRANSACTION statement:

- In interactive SQL
- Embedded in host language programs to be precompiled
- As part of a procedure in an SQL module
- In dynamic SQL as a statement to be dynamically executed

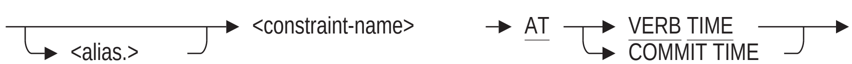
Format



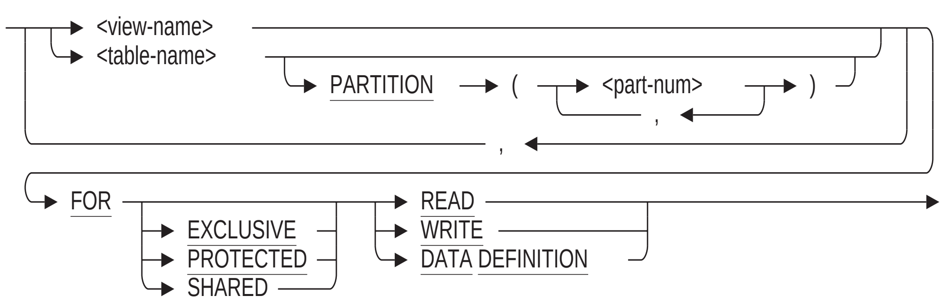
tx-options =



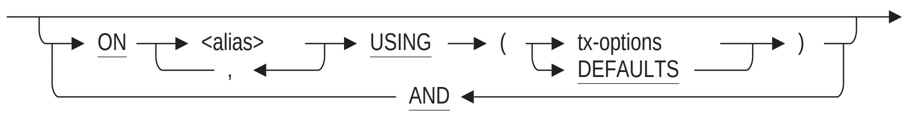
evaluating-clause =



reserving-clause =



db-txns =



Arguments

PARTITION (part-num)
Specifies the partition number for the partition to be reserved or locked. Only values in the RDB$STORAGE_MAP_AREAS table in the RDB$ORDINAL_POSITION column can be specified. SQL ignores duplicate part-num values in the reserving clause. Partitions that you do not list are reserved for shared access by default.

The PARTITION clause and the DATA DEFINITION clause are mutually exclusive; you cannot specify one if you specify the other.

Usage Notes

- The partition clause is not permitted if a table does not have a storage map, or has a vertically partitioned storage map (that is, it uses the STORE COLUMNS clause). If an index and the storage map have identical STORE clauses, then both are locked using the same list of partition numbers.
- Using the PARTITION clause requires careful database and application design. If the indexes are partitioned using different partitioning keys or different value ranges, then cross-partition updates might lead to deadlocks and other lock conflicts between concurrent update processes.
- By default, a transaction that reserves a table for EXCLUSIVE access does not reserve the LIST (segmented string) area for exclusive access. Because the LIST area is usually shared by many tables, SHARED access is assumed by default to permit updates to the other tables.

  This means that when you run an import operation or when an application updates a table reserved for EXCLUSIVE access, you might notice that the snapshot storage area (.snp) grows. This is because of the I/O to the LIST area that is performed by default when SHARED WRITE mode is in use.

However, if you attach to the database using an SQL ATTACH or IMPORT statement and you specify the RESTRICTED ACCESS clause, then all storage areas are accessed in EXCLUSIVE mode. Use this clause to eliminate the snapshot I/O and related overhead if you are performing a lot of I/O to the LIST storage areas (for example, when you are restructuring the database, or dropping a large table containing LIST OF BYTE VARYING columns and data).

Examples

Example 1 Reserving a Partition

```
SQL> -- This example locks only the second partition of
SQL> -- the EMPLOYEES table in exclusive write mode.
SQL> -- The advantage of this is that the process can insert,
SQL> --  update, or delete from this partition without writing
SQL> -- to the snapshot (.snp) file, and in general, uses fewer
SQL> --  resources for operations on the partition.
SQL> SET TRANSACTION READ WRITE
cont> RESERVING EMPLOYEES PARTITION (2) FOR EXCLUSIVE WRITE;
```

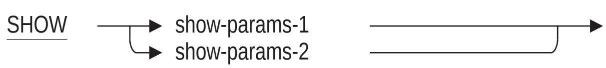
SHOW Statement

Displays information about database entities and information about the interactive SQL session.

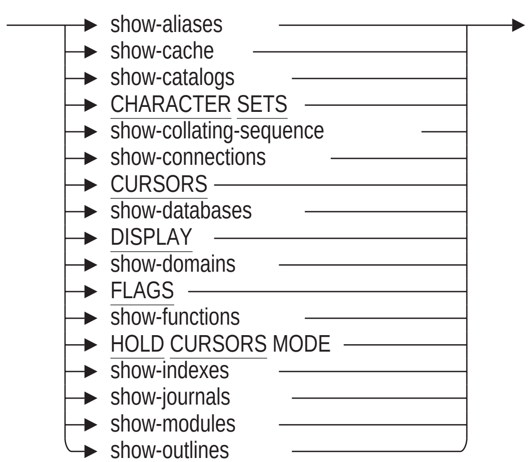
Environment

You can use the SHOW statement only in interactive SQL.

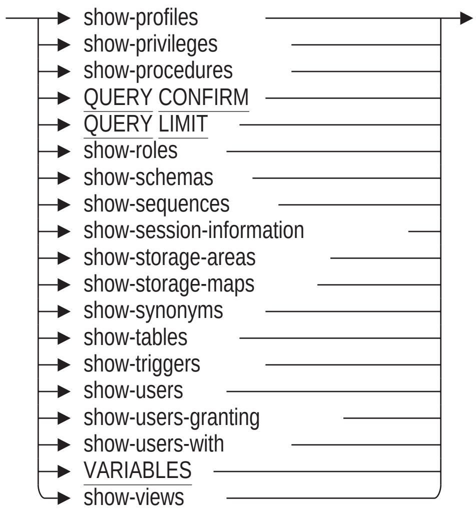
Format



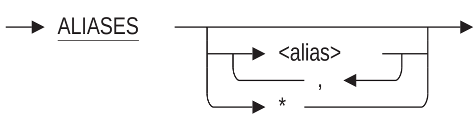
show-params-1 =



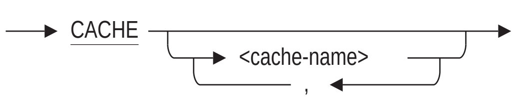
show-params-2 =



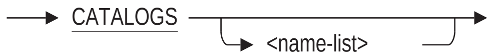
show-aliases =



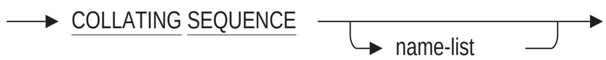
show-cache =



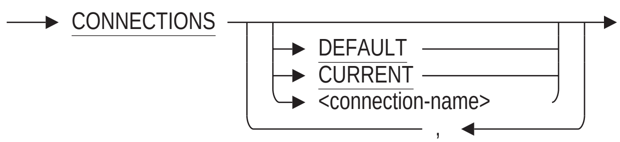
show-catalogs =



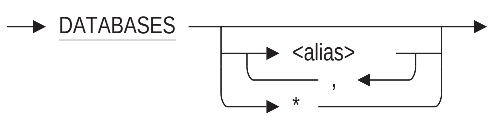
show-collating-sequence =



show-connections =

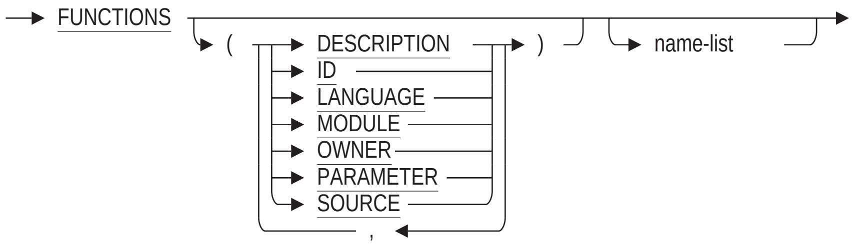


show-databases =

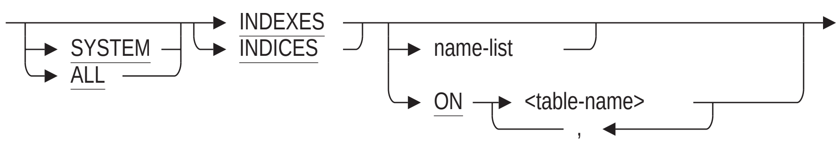


show-domains =

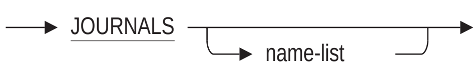
show-functions =



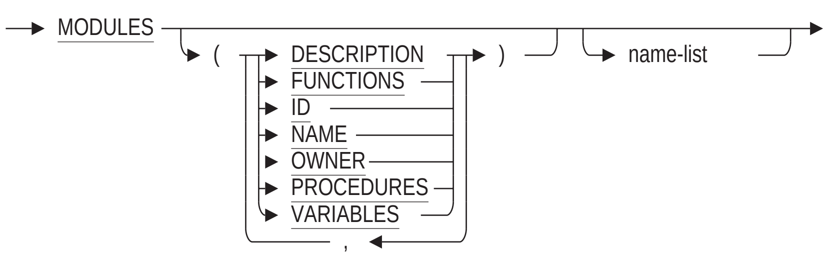
show-indexes =



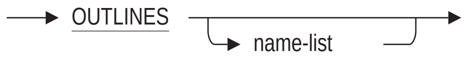
show-journals



show-modules =

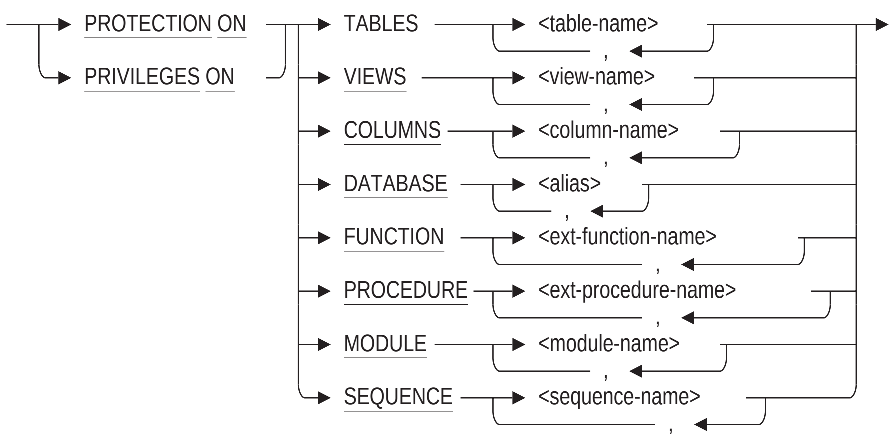


show-outlines

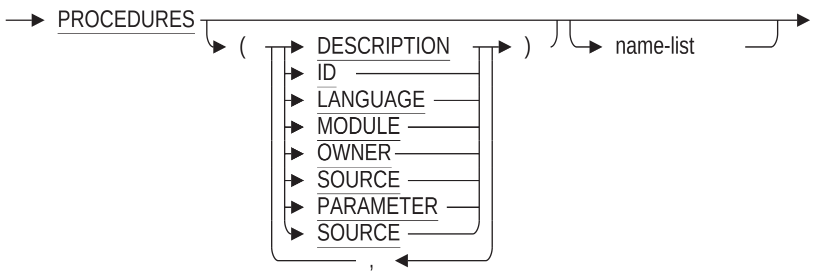


show-profiles =

show-privileges =



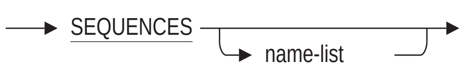
show-procedures =



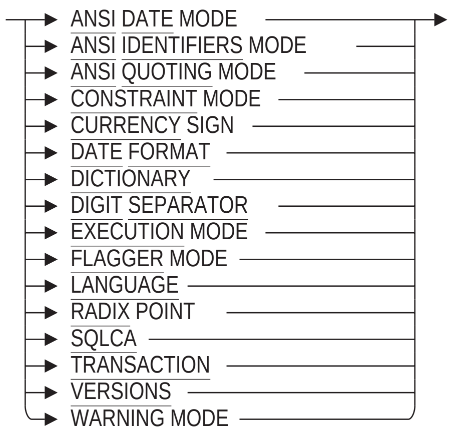
show-roles =

show-schemas =

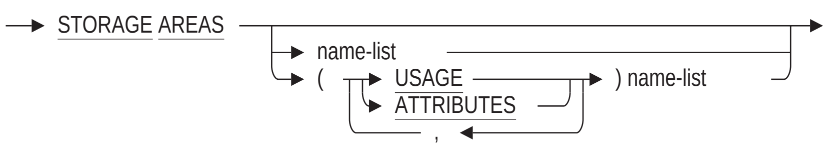
show-sequences=



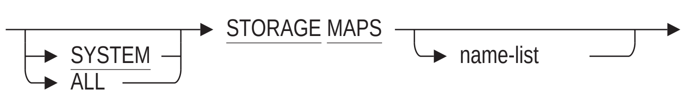
show-session-information =



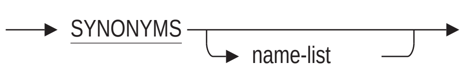
show-storage-areas =



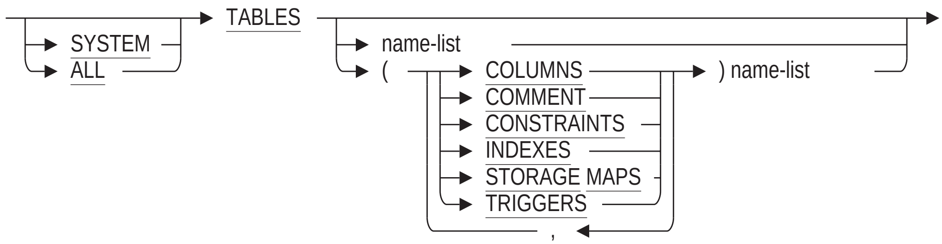
show-storage-maps =



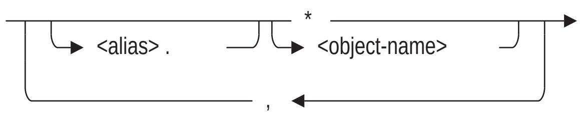
show-synonyms =



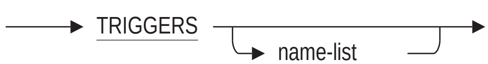
show-tables =



name-list =

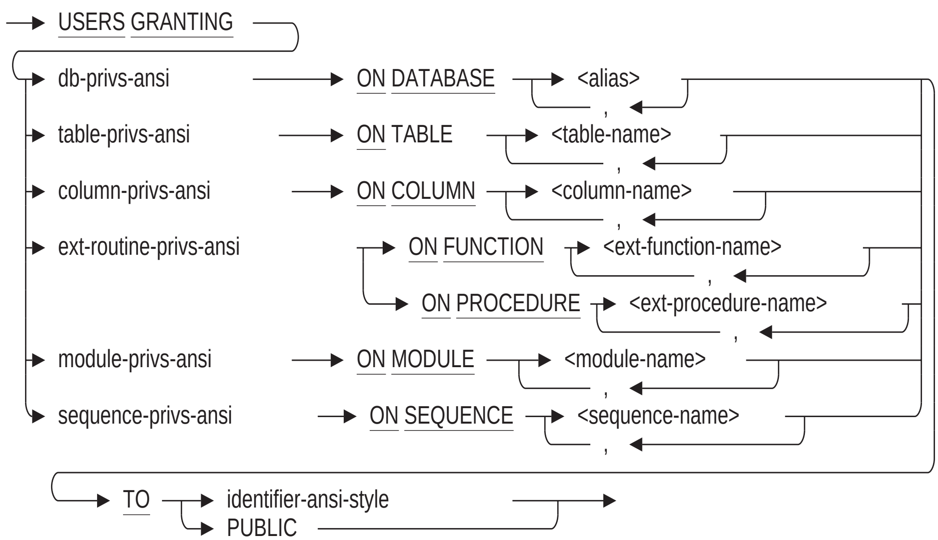


show-triggers =



show-users =

show-users-granting =



db-privs-ansi =

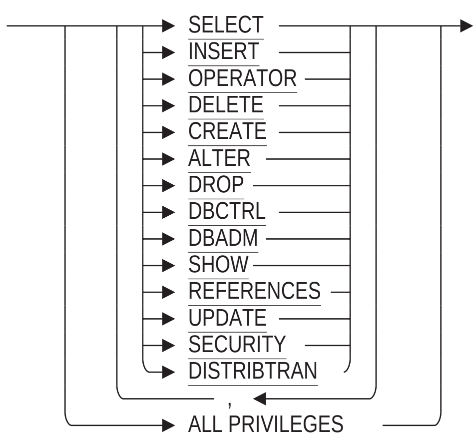
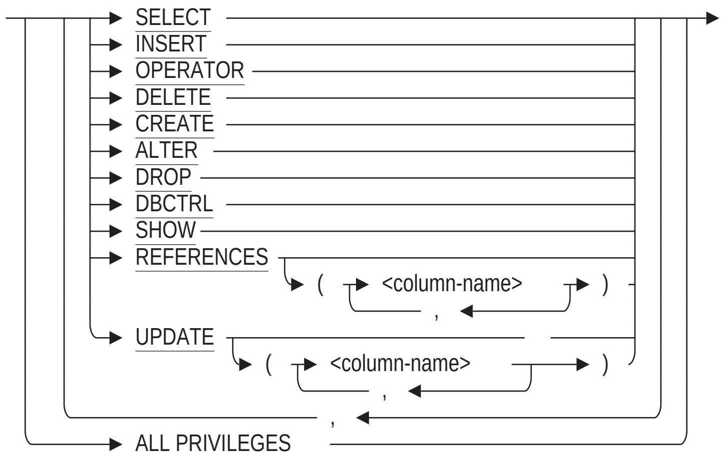


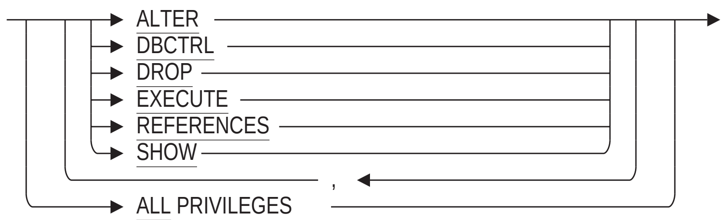
table-privs-ansi =



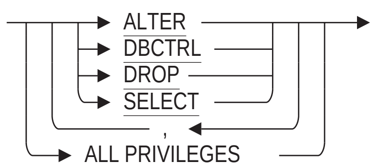
column-privs-ansi =

ext-routine-privs-ansi =

module-privs-ansi =

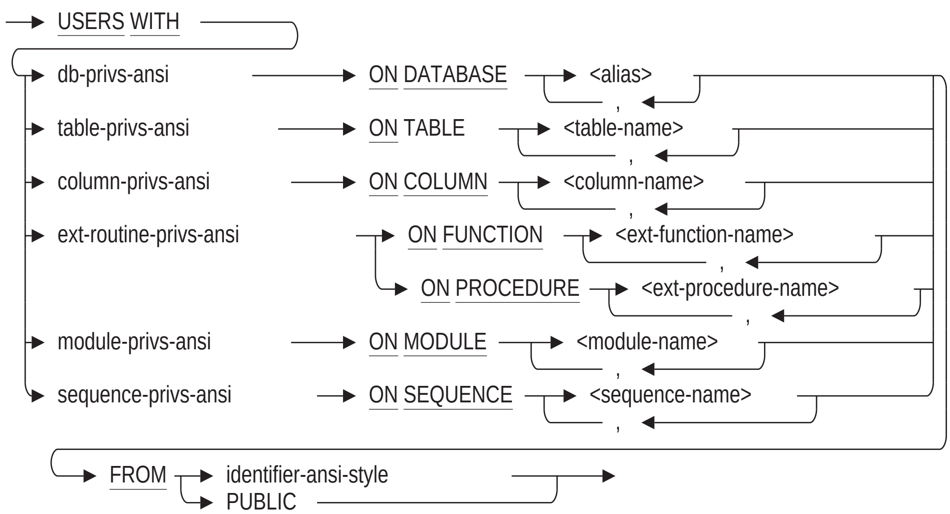


sequence-privs-ansi =

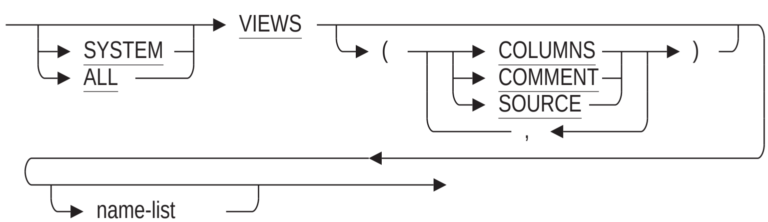


identifier-ansi-style =

show-users-with =



show-views =



Arguments

PRIVILEGES
PROTECTION

Displays the current user identifier and available access rights for the specified object.

- The SHOW PRIVILEGES statement displays the current user identifier and available access rights to the specified databases, tables, views, columns, external functions, external procedures, modules, or sequences.

  This statement displays not only the privileges that are explicitly granted to the user, but also any privileges that the user inherits from database access or the system.

  In a client/server environment, the entry shows the identifier of the client. For example, if a user attaches to a remote database using the USER and USING clauses, SQL shows the privileges for the user specified in those clauses.

  In an environment that is not client/server, such as when you attach to a local database on OpenVMS, SQL shows not only the privileges of the database user, but of the logged-on process. For example, if user heleng, with the OpenVMS privilege BYPASS, uses the USER and USING clauses to attach to the database as user rhonda, SQL shows that user rhonda has the privileges inherited from the logged-on process heleng, as well as privileges for user rhonda.
- The SHOW PROTECTION statement displays all of the entries in the access privilege set for the specified databases, tables, views, columns, external functions, external procedures, modules, or sequences.

ON SEQUENCE sequence-name

Specifies the sequence for which you want to display access privilege set information with the SHOW PRIVILEGES or SHOW PROTECTION statement. You can specify a list of sequences, but you must specify at least one item to display a list.

In an ANSI/ISO-style database, the SHOW PROTECTION statement displays which privileges have the option of being granted to other users and which privileges are without the grant option. See the SHOW USERS WITH and SHOW USERS GRANTING statements in the SHOW STATEMENT section of the *Oracle Rdb7 SQL Reference Manual* for more information about displaying privileges granted directly or indirectly to other users.

profilename

The name of the profile to be displayed. If no name is specified, the names of existing profiles will be displayed. If an asterisk (*) is used, all details of all profiles will be displayed. An optional database alias can be used to specify a database other than that of the default alias.

ROLES
Displays the definition of the specified role. SQL displays the role name, ID number, and any comments associated with the role definition.

SEQUENCES
Displays the definition of the specified sequence. SQL displays the sequence name, ID number, and the sequence attributes.

USERS
Displays the definition of the specified database user. SQL displays the database user name (such as defined by the CREATE USER statement), how the user will be authenticated (currently, only through the operating system), whether the account is locked or unlocked, and any comments associated with the user definition.

For obj-type object-name
This type of SHOW SYNONYM statement shows all the synonyms defined for a particular object. Use this format to answer the question: What alternate names can be used to reference a database object?

synonym-name
The SHOW SYNONYM statement displays all the attributes for the named synonym. An asterisk can be specified to list all the synonyms for a database alias.

obj-type
The SHOW SYNONYMS statement can show all synonyms for a certain set of database objects, such as all procedures.

Usage Notes

- If you use the ALTER TABLE statement to change the order in which columns are displayed, that ordering is also reflected when you issue a SHOW TABLE statement.
- If you issue a SHOW TABLES (CONSTRAINTS) statement, it indicates whether or not the constraint has been disabled.
- If you issue a SHOW TRIGGERS statement, it indicates whether or not the trigger has been disabled.
- Profile names are by default in uppercase. If they were defined in mixed case or with other special characters, use the SET DIALECT ’SQL92’ or ’ORACLE LEVEL1’ or ’SQL99’ or SET QUOTING RULES ’SQL92’ or ’ORACLE LEVEL1’ or ’SQL99’ statement to enable delimited identifiers. Then, use quotation marks (" ") around the name in the SHOW PROFILES statement.

- The following usage notes apply to synonyms only:
  – Synonym names are by default in uppercase. If they were defined in mixed case, or with other special characters, then use the SET DIALECT 'SQL92' or 'ORACLE LEVEL1' or 'SQL99', or SET QUOTING RULES 'SQL92' or 'ORACLE LEVEL1' or 'SQL99' statements to enable delimited identifiers. Then use quotes (" ") around the name in the SHOW SYNONYMS statement.
  – If you qualify the SHOW SYNONYMS statement with an object type different from the object type used by the synonym, then the synonym will not be displayed.
  – If neither synonym name nor asterisk (*) is provided, then a list of all synonyms will be displayed with the type of object. If the word "synonym" appears in the description, then the source of this synonym is another synonym. In this case, use SHOW SYNONYM on the source object to get more information, otherwise use the appropriate SHOW statement for the named object.
  – If an asterisk (*) or a synonym name is specified then the synonym, its comment and details about the source object are displayed.
  – If a synonym is defined for a table, view, sequence, domain, module, procedure or function, then a SHOW for that type of object will also list the defined synonyms.

Examples

Example 1 Displaying a Sequence

```
SQL> SHOW SEQUENCE EMPIDS
     EMPIDS
 Sequence Id: 3
 Initial Value: 1
 Minimum Value: 1
 Maximum Value: 9223372036854775787
 Next Sequence Value: 1
 Increment by: 1
 Cache Size: 20
 Order
 No Cycle
 No Randomize
 Comment:       Sequence for employee IDs.
```

Example 2 Displaying a Role

```
SQL> SHOW ROLE SECRETARY
     SECRETARY
 Identified Externally
 Comment:       Role for the secretarial staff
```

Example 3 Displaying a User

```
SQL> SHOW USER NSTEWART
     NSTEWART
 Identified Externally
 Account Unlocked
 Comment:       Nicholas Stewart
```

Example 4 Show Details of One Profile

```
SQL> SHOW PROFILE
Profiles in database with filename SQL$DATABASE
     DECISION_SUPPORT
SQL> SHOW PROFILE DECISION_SUPPORT
     DECISION_SUPPORT
 Comment:       limit transactions used by report writers
     Transaction modes (read only, no read write)
SQL> ALTER PROFILE DECISION_SUPPORT
cont>   default transaction read only;
SQL> SHOW PROFILE DECISION_SUPPORT
     DECISION_SUPPORT
 Comment:       limit transactions used by report writers
     Default transaction read only
     Transaction modes (read only, no read write)
SQL>
```

Example 5 Show the Use of Delimited Identifiers for Mixed-Case Names

```
SQL> CREATE PROFILE "Decision_Support"
cont> COMMENT IS ’limit transactions used by report writers’
cont> TRANSACTION MODES (NO READ WRITE, READ ONLY);
SQL> SHOW PROFILE
Profiles in database with filename SQL$DATABASE
     Decision_Support
SQL> SHOW PROFILE Decision_Support
 No Users found
SQL> SHOW PROFILE "Decision_Support"
```

```
     Decision_Support
 Comment:       limit transactions used by report writers
     Transaction modes (read only, no read write)
```

Example 6 Displaying Synonyms

```
SQL> SHOW SYNONYMS
Synonyms in database with filename SQL$DATABASE
     C_SAL                            View                CURRENT_SALARY
     E                                Table synonym       EMPS
     EMPS                             Table               EMPLOYEES
     ID_NUMBER                        Domain              ID_DOM

SQL> SHOW SYNONYMS ID_NUMBER
 ID_NUMBER
 for domain  ID_DOM
 Comment:       support the old name for this domain

SQL> SHOW VIEWS
User tables in database with filename SQL$DATABASE
     CURRENT_INFO                     A view.
     CURRENT_JOB                      A view.
     CURRENT_SALARY                   A view.
     C_SAL                            A synonym for view CURRENT_SALARY
```

SIGNAL Control Statement

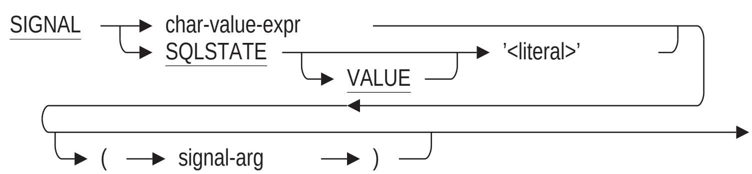
Passes the signaled SQLSTATE status parameter, and optionally a secondary message, back to the application or SQL interface and terminates the current routine and all calling routines.

Environment

You can use the SIGNAL statement in a compound statement:

- In interactive SQL
- Embedded in host language programs to be precompiled
- As part of a procedure in an SQL module
- In dynamic SQL as a statement to be dynamically executed

Format



Arguments

SQLSTATE VALUE literal
Specifies the SQLSTATE status parameter. Any provided value is converted to a CHAR(5) value that is passed to SIGNAL. This is provided to conform with the SQL / PSM standard.

signal-arg
Specifies a value expression. The specified value is converted to a CHARACTER(80) CHARACTER SET UNSPECIFIED string and returned as a secondary message to the client application. If the value expression converts to a character string longer than 80 characters, it is truncated.

You can use the sql_get_error_text routine to extract the signal-arg text in an application.

Usage Notes

No new usage notes.

Examples

Example 1 Specifying a Secondary Error

```
SQL> BEGIN
SQL> SIGNAL SQLSTATE 'RR000' (' Compound Statement Failed');
cont> END;
%RDB-E-SIGNAL_SQLSTATE, routine "(unnamed)" signaled SQLSTATE "RR000"
-RDB-I-TEXT,  Compound Statement Failed
```

START TRANSACTION Statement

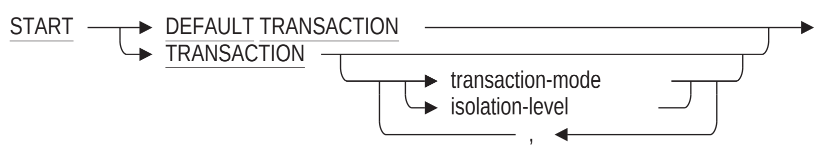
The START TRANSACTION statement is specified by the SQL:1999 standard. The standard syntax has been extended for Oracle Rdb to support the default transaction defined in a user profile.

Environment

You can use the START TRANSACTION statement:

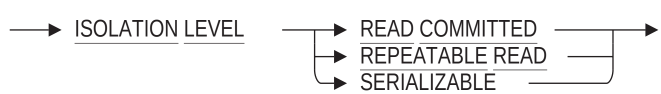
- In interactive SQL
- Embedded in host language programs
- As part of a procedure in an SQL module or other compound statement
- In dynamic SQL as a statement to be dynamically executed

Format



transaction-mode =

isolation-level =



Usage Notes

- The START TRANSACTION statement is similar to the SET TRANSACTION statement in operation. That is, you can specify READ WRITE or READ ONLY transaction modes as well as various isolation levels.
- The transaction-mode and isolation-level clauses may appear only once in any START TRANSACTION statement.
- This statement does not support BATCH UPDATE mode, as this is an Oracle Rdb extension and, therefore, is only supported by SET and DECLARE TRANSACTION statements.
- The alternate forms for the transaction mode READ_ONLY and READ_WRITE (single keywords) that are supported by the SET and DECLARE TRANSACTION statements are not supported for this statement.
- Oracle Rdb has extended the START TRANSACTION statement and allows all transaction options to be omitted. If the transaction-mode is omitted, it defaults to READ WRITE. If the isolation-level is omitted, it defaults to ISOLATION LEVEL SERIALIZABLE. Therefore, if all options are omitted, the transaction defaults to READ WRITE ISOLATION LEVEL SERIALIZABLE.
- If more than one database is currently attached, a transaction spanning all databases will be started with the specified or default attributes.
- If the keyword DEFAULT is used, the user-specific default transaction is started. This default is defined in the profile for the current session user. If none is specified, a READ ONLY transaction will be started.

  ```
  SQL> CREATE PROFILE READ_USERS
  cont>   DEFAULT TRANSACTION READ ONLY WAIT 10;
  SQL> ALTER USER JONES PROFILE READ_USERS;
  ```

  The START DEFAULT TRANSACTION will start a READ ONLY WAIT 10 transaction for JONES.

Examples

Example 1 Starting a Default Transaction in a Multistatement Procedure or as a Single Statement

```
SQL> START DEFAULT TRANSACTION;
SQL>
SQL> BEGIN
cont> COMMIT;
```

```
cont> START DEFAULT TRANSACTION;
cont> END;
SQL>
SQL> ROLLBACK;
```

Example 2 Starting Several Variations of the START TRANSACTION Statement

```
SQL> START TRANSACTION READ WRITE,
cont>   ISOLATION LEVEL READ COMMITTED;
SQL> COMMIT;
SQL>
SQL> -- Defaults to serializable
SQL> START TRANSACTION READ WRITE;
SQL> COMMIT;
SQL>
SQL> -- Defaults to read write
SQL> START TRANSACTION ISOLATION LEVEL READ COMMITTED;
SQL> ROLLBACK;
SQL>
SQL> -- Defaults to read write serializable
SQL> START TRANSACTION;
SQL>
SQL> BEGIN
cont> COMMIT;
cont> START TRANSACTION
cont>     ISOLATION LEVEL READ COMMITTED,
cont>     READ WRITE;
cont> END;
SQL> COMMIT;
```

TRACE Control Statement

Writes values to the trace log file after the trace extended debug flag is set. The TRACE control statement lets you specify multiple value expressions. It stores a value in a log file for each value expression it evaluates.

SQL turns on trace logging only if the logical name RDMS$DEBUG_FLAGS is defined to be *Xt* (the letter *X* must be an uppercase letter and the letter *t* must be in lowercase) or if the SET FLAGS 'TRACE' statement has been executed.

Trace logging can help you debug complex multistatement procedures.

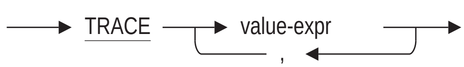
Environment

You can use the TRACE control statement in a compound statement:

- In interactive SQL
- Embedded in host language programs to be precompiled
- As part of a procedure in an SQL module
- In dynamic SQL as a statement to be dynamically executed

Format

trace-statement =



Arguments

No new arguments.

Usage Notes

- If the TRACE statement is activated by the RDMS$DEBUG_FLAGS "Xt" (or by the SET FLAGS statement), then queries in the TRACE statement are merged into the query outline for the procedure. Example 1 in the Examples section shows a query outline that contains one query when the TRACE statement is disabled.

- If the query outline is generated with TRACE enabled, then two queries appear; the first is for the subquery in the TRACE statement and the other is for the singleton SELECT statement. See Example 2 in the Examples section.

  If this second query outline is used at run time with the TRACE statement disabled, then it cannot be applied to the query. See Example 3 in the Examples section. Because the outline was created with compliance optional, the query outline is abandoned and a new strategy is calculated. If compliance is mandatory, then the query fails.

  If any TRACE statement contains a subquery, then Oracle Corporation recommends using two query outlines (if any are required at all), with different modes in order to run the query with and without TRACE enabled. That is, when TRACE is enabled, define RDMS$BIND_OUTLINE_MODE to match the TRACE enabled query outlines.

  ```
  $ DEFINE RDMS$DEBUG_FLAGS "Xt"
  $ DEFINE RDMS$DEBUG_FLAGS_OUTPUT TRACE.DAT
  $ DEFINE RDMS$BIND_OUTLINE_MODE 10
  ```

  Alternatively, use the SET FLAGS statement, which allows the TRACE flag to be enabled and the MODE established from within an interactive session or through dynamic SQL. This method allows the query to be run with TRACE enabled or disabled.

- Use the COALESCE function to format NULL expressions. For example, TRACE COALESCE(LAST_NAME, ’NULL’);.

Examples

Example 1 Generating a Query Outline When the TRACE Statement Is Disabled

```
SQL> DECLARE :LN CHAR(40);
SQL>
SQL> BEGIN
cont> TRACE 'Jobs Held: ',
cont>     (SELECT COUNT(*)
cont>         FROM JOB_HISTORY
cont>         WHERE EMPLOYEE_ID = '00201');
cont> SELECT LAST_NAME
cont>     INTO :LN
cont>     FROM EMPLOYEES
cont>     WHERE EMPLOYEE_ID = '00201';
cont> END;
```

```
-- Oracle Rdb Generated Outline : 28-MAY-1997 16:48
create outline QO_A17FA4B41EF1A68B_00000000
id 'A17FA4B41EF1A68B966C1A0B083BFDD4'
mode 0
as (
  query (
-- Select
    subquery (
      EMPLOYEES 0     access path index       EMPLOYEES_HASH
      )
    )
  )
compliance optional     ;
SQL>
```

Example 2 Generating a Query Outline When the TRACE Statement Is Enabled

```
SQL> DECLARE :LN CHAR(40);
SQL>
SQL> BEGIN
cont> TRACE 'Jobs Held: ',
cont>     (SELECT COUNT(*)
cont>         FROM JOB_HISTORY
cont>         WHERE EMPLOYEE_ID = '00201');
cont> SELECT LAST_NAME
cont>     INTO :LN
cont>     FROM EMPLOYEES
cont>     WHERE EMPLOYEE_ID = '00201';
cont> END;
-- Oracle Rdb Generated Outline : 28-MAY-1997 16:48
create outline QO_A17FA4B41EF1A68B_00000000
id 'A17FA4B41EF1A68B966C1A0B083BFDD4'
mode 0
as (
  query (
-- Trace
    subquery (
      JOB_HISTORY 0   access path index       JOB_HISTORY_HASH
      )
    )
  query (
-- Select
    subquery (
      EMPLOYEES 0     access path index       EMPLOYEES_HASH
```

```
      )
    )
  )
compliance optional     ;
~Xt: Jobs Held: 4
SQL>
```

Example 3 Using an Outline with Tracing Enabled That Was Created with Tracing Disabled

```
SQL> DECLARE :LN CHAR(40);
SQL>
SQL> BEGIN
cont> TRACE ’Jobs Held: ’,
cont>     (SELECT COUNT(*)
cont>         FROM JOB_HISTORY
cont>         WHERE EMPLOYEE_ID = ’00201’);
cont> SELECT LAST_NAME
cont>     INTO :LN
cont>     FROM EMPLOYEES
cont>     WHERE EMPLOYEE_ID = ’00201’;
cont> END;
~S: Outline QO_A17FA4B41EF1A68B_00000000 used
~S: Outline/query mismatch; assuming JOB_HISTORY 0 renamed to EMPLOYEES 0
~S: Full compliance with the outline was not possible
Get     Retrieval by index of relation EMPLOYEES
  Index name  EMPLOYEES_HASH [1:1]       Direct lookup
```

WHILE Control Statement

Allows the repetitive execution of one or more SQL statements in a compound statement based on the truth value of a predicate.

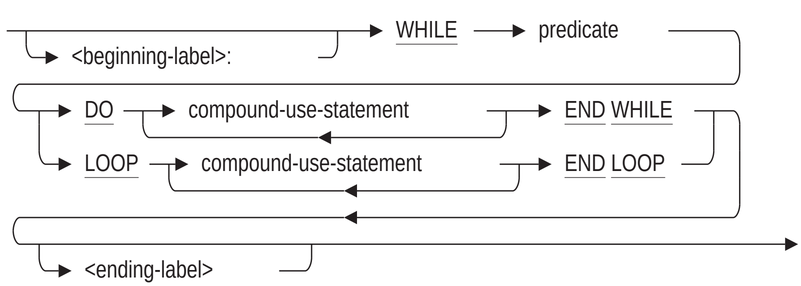
Environment

You can use the WHILE statement in a compound statement:

- In interactive SQL
- Embedded in host language programs to be precompiled
- As part of a procedure in an SQL module
- In dynamic SQL as a statement to be dynamically executed

Format

while-statement=



Arguments

beginning-label:
Assigns a name to a control loop. A beginning label used with the LEAVE statement lets you perform a controlled exit from the WHILE loop. If you include an ending label, it must be identical to its corresponding beginning label. A beginning label must be unique within the procedure containing the label.

WHILE predicate

Specifies a search condition that controls how many times SQL can execute a compound statement.

SQL evaluates the WHILE search condition. If it evaluates to TRUE, SQL executes the associated sequence of SQL statements (called a compound statement). If SQL does not encounter an error exception, control returns to the WHILE clause at the top of the loop for subsequent evaluation. Each time the search condition evaluates to TRUE, the WHILE-LOOP or WHILE-DO statement executes the SQL statements embedded within its LOOP ... END LOOP or DO ... END WHILE block. If the search condition evaluates to FALSE or UNKNOWN, SQL bypasses the LOOP ... END LOOP or DO ... END WHILE block and passes control to the next statement.

DO

Marks the start of a control loop. A WHILE statement lets you execute the associated sequence of SQL statements, called a **compound statement**. After SQL executes the statements within the DO ... END WHILE loop, control returns to the DO keyword at the top of the loop for subsequent statement execution. Looping occurs until SQL encounters an error exception or executes a LEAVE statement. In either case, SQL passes control out of the DO block to the statement immediately after the WHILE statement.

LOOP

Marks the start of a control loop. A WHILE statement enables you to execute the associated sequence of SQL statements, called a **compound statement**. After SQL executes the statements within the loop, control returns to the LOOP keyword at the top of the loop for subsequent statement execution. Looping occurs until SQL encounters an error exception or executes a LEAVE statement. In either case, SQL passes control out of the LOOP block to the statement immediately after the WHILE statement.

compound-use-statement

Identifies the SQL statements allowed in a compound statement block. See the Compound Statement for the list of valid statements.

END WHILE ending-label

Marks the end of a DO control loop. If you choose to include the optional ending label, it must match exactly its corresponding beginning label. An ending label must be unique within the procedure in which the label is contained.

The optional ending-label argument makes multistatement procedures easier to read, especially in very complex multistatement procedure blocks.

END LOOP ending-label

Marks the end of a LOOP control loop. If you choose to include the optional ending label, it must match exactly its corresponding beginning label. An ending label must be unique within the procedure in which the label is contained.

The optional ending-label argument makes multistatement procedures easier to read, especially in very complex multistatement procedure blocks.

Usage Notes

- Although the DO ... END WHILE and LOOP ... END LOOP are semantically equivalent, the DO ... END WHILE syntax conforms to the ANSI/ISO SQL/PSM standard.

Examples

Example 1 Using the While Statement to Count Substrings

```
SQL> DECLARE :SUB_STR CHAR;
SQL> DECLARE :SRC_STR CHAR(50);
SQL> BEGIN
cont>   SET :SUB_STR='l';
cont>   SET :SRC_STR='The rain in Spain falls mainly on the plain';
cont> END;
SQL> SET FLAGS 'TRACE';
SQL> BEGIN
cont>-- This procedure counts the occurrence of substrings
cont>   DECLARE :STR_COUNT INTEGER=0;
cont>   DECLARE :CUR_POS INTEGER = POSITION (:SUB_STR IN :SRC_STR);
cont>   WHILE :CUR_POS >0 DO
cont>      SET :STR_COUNT=:STR_COUNT + 1;
cont>         SET :CUR_POS = POSITION (:SUB_STR IN :SRC_STR FROM :CUR_POS + 1);
cont>   END WHILE;
cont>     TRACE 'FOUND ', :STR_COUNT, ' OCCURRENCES OF "', :SUB_STR, '"';
cont> END;
~Xt: Found 4          occurrences of "l"
```

4

Using Row Caching on OpenVMS

On OpenVMS, Oracle Rdb uses buffers to temporarily store database pages during read and update operations. When you create or modify a database, you can set up buffers for database pages in either of the following ways:

- Local buffers

  Database users have their own set of private local database page buffers. Data of interest is read from disk into a local database page buffer. Local buffers are not shared among users. Sharing occurs only when a database page is written back to disk and another user retrieves that database page. The sharing is done at the physical page level and can be I/O intensive.

- Global buffers

  Database users on the same system share a common set of global database page buffers that reside in global memory. Database pages that are read from disk by one user can be seen directly by another user. Little or no I/O is needed to share global buffers; however, sharing data is still done at the level of database page buffers. A database page buffer has a fixed size across all storage areas in the database. The amount of data in a database page buffer that is of interest to multiple users may be small compared to its overall size. Although this model may be more efficient than using local buffers, there are better ways to share data among users.

Oracle Rdb offers a feature called row caching to enhance the performance of memory buffers. Because row caching works with a cache of rows, you can use it in conjunction with local or global database page buffers. Consider, however, that when you use both global buffers and a row cache, you can have two copies of data consuming your global memory: one copy in the row cache and one in a global buffer. Note also that row caches are not designed to be an *in-memory* database. As its name implies, a **row cache** is a set of database rows that reside in memory between the users and the rest of the database rows on disk. Data rows, system records, and hashed and sorted index nodes can be cached. Access to a row in a row cache is through its logical database key (dbkey).

All processes attached to a database share a pool of row occurrences that reside in shared memory row caches. No disk I/O is needed to share a row in a row cache. Only the rows of interest, not the physical pages, are kept in shared memory, thereby increasing the use of shared memory. In addition, you can create many row caches, each with its own row size. Row caches can be used to efficiently store rows of specific sizes from specified tables. The Oracle Rdb implementation of row caching gives you the option to specify portions of row caches to occupy process private virtual memory, shared global pagefile sections on OpenVMS systems, or shared physical main memory. Oracle Rdb row caching also allows you to use very large memory (VLM) on OpenVMS Alpha systems. Subsequent sections provide more details on each of these options.

The row caching feature is designed to improve performance through reduced I/O operations by finding rows of interest in the row cache instead of accessing them on disk. The greater number of times the data is located in the row cache, the more useful the cache is, and better overall performance results.

The next section describes how row caching works with basic Oracle Rdb database functions.

4.1 Database Functions Using Row Cache

The following list describes how common database operations use the row caching feature.

- Fetching data

  When you request a row from a database, Oracle Rdb first checks to see if the requested row is located in a row cache. If the row is in a row cache, the row is retrieved from the cache. If the row is not in a cache, Oracle Rdb checks the page buffer pool. If the row is not in the page buffer pool, Oracle Rdb performs a disk I/O operation to retrieve the row. The requested row is then inserted into the row cache, if possible.

- Storing data

  When a new row is stored in the database, Oracle Rdb may perform a disk I/O operation to find space for the new row and get a dbkey for the row. Once space has been reserved on a database page, Oracle Rdb checks for a row cache in which to put the new row. The new row is inserted into a row cache, if possible.

- Modifying data

If a modification to a row in a cache causes the row to grow (because the modification replaces a null value, for example), then the database page must be modified to reserve additional space for that row. If the database page does not have room for the modified row, resulting in fragmentation, then the row is deleted from the cache. If the modification keeps the row the same size or makes it smaller, then the modified row remains in the cache and no database page is accessed. This means that the unused space on the page is not reclaimed and hence is not immediately available for reuse. Compressed rows and indexes that are modified are more likely to require database access than uncompressed ones.

- Deleting data

  If the row is in a row cache, Oracle Rdb sets the length of the row to zero to erase it. It is not erased from the database page on disk immediately. Therefore, the deleted space is not reusable immediately.

- When snapshots are enabled

  During a read-only transaction, Oracle Rdb first checks to see if the row is in a row cache. If the row is found and is visible to the transaction, the row is returned from the row cache and no disk I/O operation is necessary. If the row is not visible, Oracle Rdb must find the visible version of this row in the snapshot file. Information stored in the row cache, however, can shorten the search and thereby reduce I/O operations to the snapshot file.

  During a read/write transaction that is performing an update, Oracle Rdb writes the before-image of the data to the snapshot file. Oracle Rdb writes the before-image information to the snapshot file each time a row in the user's row cache working set is modified. If a row in the working set list is replaced by a new row of interest, and then is remodified later in the transaction, the before-image information is written back to the snapshot file when the row reenters the working set.

  Global and local buffers use the least recently used (LRU) replacement strategy for database pages. Row caching uses a modified form of the LRU replacement strategy. Each database user can protect the last 10 rows he or she accessed. This group of rows is referred to as a **working set**. Rows that belong to a working set are considered to be **referenced** and are not eligible for row replacement.

  During a read/write transaction that performs a delete operation, the processing is the same as described in the previous paragraphs.

4.2 Writing Modified Rows to Disk

With row caching, many data modifications are performed on the copy of the data in memory. Therefore, Oracle Rdb must have a way to write these rows to storage on disk.

The following list describes the ways that modified rows can be written back to the database page on disk.

- If the page on which a modified row resides is in the user's buffer pool and is already locked by the user when the update to that row must be recorded in the row cache, then the update is made to the row in the cache and on the database page.

  In this case, the row cache entry is not considered to be marked or modified. This situation occurs when a transaction is committed or when a row is flushed from a row cache.

- During an undo operation, the before-image of each modified row is placed on the database page.

  An **undo** operation occurs as part of an aborted SQL statement, transaction rollback, or database recovery of a terminated user's process.

- During a **redo** operation, the after-image of each modified row is stored on the database page only if the database is recovering from a node failure. If the database is recovering from a process failure, no redo is done for in-memory row cache modifications because the row cache memory is still valid and intact. (Changes made to database pages are still redone.)

- During a row cache checkpoint operation, all modified rows (or all rows) from the row caches are written to disk storage.

  This is the most common method of writing updated rows back to disk storage.

- During a row cache sweep operation, a set of modified rows is written back to the database from the row cache. After the rows are written back to disk, the space they occupied is considered selectable for reuse.

  A row cache sweep operation is initiated when a user process tries to insert rows into a row cache and finds no free space available.

4.3 Row Cache Checkpointing and Sweeping

Checkpointing and sweeping operations are critical in performing the operations necessary to write modified, committed rows back to disk from a row cache. The row cache server (RCS) process performs these tasks. There is one RCS process per database. Any failure of the RCS process forces the shutdown of the entire database.

To monitor the status of rows in a row cache, Oracle Rdb maintains a modification flag for every row in a cache to indicate which rows have been modified. The modification flags are shown in the following table:

Modification Flag	Meaning
Marked	The row has been modified in the row cache only. If this modification remains only in the row cache at the time the transaction is committed, then this marked flag indicates this row in the row cache is not reflected in the database.
Hot	The marked row has been modified since the last checkpoint.
Cold	The marked row has not been modified since the last checkpoint.

The RCS process performs three types of operations:

- Synchronous operations in which the requester is waiting for the operation to complete

  The following are operations of this type:

  - The RCS process checkpoint operation that is part of an AIJ fast-commit checkpoint

    For example, if the RMU Checkpoint command with the Wait qualifier is issued, then the requester will wait for the RCS process to complete its checkpoint.

  - A checkpoint to the database for all row caches before certain database utility operations can begin

- Row cache checkpoint operations

  Checkpointing is a repetitive, time-driven event that writes rows from all row caches back to disk storage. The RCS process writes data to a cache backing file (.rdc) or directly to the database for each cache, depending on how the row cache was defined. The time interval at which a checkpoint occurs is also programmable. When the last user detaches from the database, the RCS process performs a final checkpoint operation to the database (never to the cache backing files). See Section 4.8.2.1 for more details.

- Row cache sweep operations

  Sweeping is done to make space available in a particular row cache. When a transaction requests space and none is available, the RCS process sweeps marked rows back from the particular row cache to the database. It also resets row cache reference counts if your database has experienced some user process failures. This creates free memory for subsequent transactions to insert rows into each cache. This may never be necessary if checkpointing is done at appropriate intervals. See Section 4.8.2.3 for more details on sweeping.

The RCS process selects work requests based on their priority; synchronous operations are checked first, then checkpoints, followed by sweep operations.

If a database is opened manually, the RCS process is started as part of the open operation. If a database is opened automatically, the RCS, by default, is started when a row cache is referenced for the first time.

When the last user disconnects from the database (with the database open setting set to automatic) or when the database is closed manually, the RCS process performs a final checkpoint to the database. When this operation completes, all marked rows have been written back to the database. The RCS process writes out its checkpoint information to indicate that backing files are no longer needed if there is a need to recover from a node failure. At this time, the cache backing files, if any, are deleted by default. If you want to preserve the backing files and have them reused at database startup, define the logical name RDM$BIND_RCS_KEEP_BACKING_FILES to 1.

Details of the RCS actions can be seen by creating an RCS process log file. Before opening the database, define the RDM$BIND_RCS_LOG_FILE system logical name to indicate the device, directory, and file name of the RCS process log file that you want to create. If no device and directory are specified, the RCS log file is created in the same directory as that which contains the database root file.

4.4 Node and Process Failure Recovery

The following sections describe how the row cache feature interacts with node and process failure recovery.

To understand how database recovery works with row caches, you should understand the interactions that occur when writing to row caches, writing to the recovery-unit journal (.ruj) files, and writing to the after-image journal (.aij) files.

This interaction is identical to the interactions that occur among database page buffers, RUJ journaling, and AIJ journaling. For more information, see the *Oracle Rdb7 Guide to Database Performance and Tuning*.

The AIJ fast commit feature is a prerequisite for enabling row caching. This means that updates to the database are not flushed back to the database pages at the time a transaction is committed. In the case of row caching, the modified rows reside in the in-memory row caches. However, all after-image rows (updated rows) must be flushed to the .aij file when the transaction is committed. In the event of a failure, the committed, updated rows can be reapplied to the database from the .aij file.

Recovery-unit journaling is critical to ensuring that rows can be returned to their previous state if either an SQL statement or transaction rolls back or aborts abnormally. The before-image of a row must be preserved before any modification is made to a row on a database page or in a row cache. Before-images are placed in an in-memory RUJ buffer. When that buffer becomes full, or when a modified page or modified row cache entry is put back, the RUJ information is first written synchronously to the .ruj file. For a database without row caches, this means that the write I/O to the .ruj file must be performed before a database page containing a modified row can be written to disk.

With row caches, Oracle Rdb frequently modifies only pages in memory, not database pages. The requirement for RUJ information to be written before a modification is put back into the row cache still exists. Writing synchronous I/Os to the .ruj file before modifying in-memory row caches does not make sense. Oracle Rdb minimizes this behavior in two ways:

- A modification to a row cache entry is first done in a local copy. Only when this local copy of the row must be flushed back to the row cache is the RUJ information written out.
- The RUJ buffer resides in a systemwide, shared memory global section that is visible to the database recovery process. Therefore, the before-image rows do not have to be written to the .ruj file unless an uncommitted modification to a database page (a store or a large modify operation) is forced to disk or the RUJ buffer overflows.

The global section created for the RUJ buffers is approximately 16 pages for each allowed database user. One global section is created for each database that has row caching enabled. If you must disable this optimization (due to insufficient memory resources) for databases with row caching enabled, define the logical name RDM$BIND_RUJ_GLOBAL_SECTION_ENABLED to 0 in the system logical name table.

You must increase several OpenVMS system parameters, as follows:

- GBLSECTIONS

  Increase by the maximum number of Oracle Rdb databases open at one time on the system.

- GBLPAGES

  Increase by 256 times the maximum number of users for each database open at one time on the system.

- GBLPAGFIL

  Increase by 16 times the maximum number of users for each database open at one time on the system.

No additional virtual memory is consumed for database users when the RUJ global buffers optimization is enabled; each user process continues to use the same amount of virtual memory (256 blocks) as when the optimization is not enabled.

Databases that do not have row caching enabled do not have the optimization enabled for the RUJ buffer in a global section.

4.4.1 Process Failure

When a process terminates abnormally, Oracle Rdb activates a database recovery (DBR) process to recover the work done by the terminated user. The DBR process first performs transaction REDO, reapplying committed transactions' modifications to the database pages that had only been written to the .aij file back to the database. Because the row cache memory is still intact, in-memory row cache changes do not have to be redone during REDO. The DBR process then proceeds to undo the user's outstanding transaction. If the RUJ systemwide process buffers are enabled, the DBR process first writes the current RUJ buffer to the .ruj file. It then recovers the .ruj file by placing the before-image of each row back on the database page. If the dbkey for that row is also found in a row cache, the before-image is placed back into the row cache, too.

4.4.2 Node Failure

There are several events that constitute a node failure to Oracle Rdb:

- A machine or operating system fails.
- The Oracle Rdb monitor process terminates unexpectedly.
- The Oracle Rdb row cache server (RCS) process terminates unexpectedly.
- An Oracle Rdb DBR process terminates unexpectedly.

- The RMU Monitor Stop command is issued with the Abort=Delprc qualifier.
- The RMU Close command is issued with the Abort=Delprc qualifier.

All of these events cause all access to an Oracle Rdb database to cease immediately. Recovery from a node failure is deferred until the next time a database attach occurs or the database is opened. Even if the RMU Open command with the Row_ Cache=Disabled qualifier is executed next, it initiates recovery from the node failure. It will not create nor populate the in-memory row caches during the recovery. Once recovery has finished, no row caches will be active while the database stays open in this manner.

Oracle Rdb has several schemes for recovering a database after a node failure. For a database without row caching enabled and without global buffers enabled, Oracle Rdb recovers from a node failure by creating one database recovery process (DBR) for each abnormally terminated user, and these database recovery processes recover the database in parallel. For a database without row caching enabled but with global buffers enabled, Oracle Rdb recovers one database user at a time by creating one DBR process at a time. For a database with row caching enabled, Oracle Rdb creates one DBR process, and that process performs recovery for all users.

If a node failure occurs, a database with row caching enabled is recovered when the DBR process performs the following steps:

1. Recovers the backing files. For each row cache that is checkpointed to a backing file, the DBR process:
   a. Reads each row from the backing file and records the dbkey for each of these rows into a row cache dbkey list.
   b. If the row has been updated (marked), then the DBR process writes this row back to the appropriate database page.
   c. Inserts this row into the empty row cache in shared memory. If the database is opened with row caching disabled or if the system logical name RDM$BIND_DBR_UPDATE_RCACHE is defined to 0, then the row caches are not repopulated from the backing files.
2. Performs a redo operation from the oldest user checkpoint. This includes the RCS process checkpoint when the RCS process last checkpointed the row caches.
   – For each transaction rolled back, the DBR process discards the updates.
   – For each transaction committed, the DBR process reapplies those updates to the database pages.

Note that all committed transactions since the oldest checkpoint are applied, not just all committed transactions for the users who were active at the time of the node failure.

- If the DBR is repopulating the row caches and a dbkey is found in the row cache dbkey list, then this occurrence replaces the current one in the row cache. If a row in a mixed format area is erased, it is removed from the row cache and its dbkey is removed from the dbkey list. This is necessary to prevent the physical dbkey that may be reused for a different table or index from being placed in the prior occurrence's row cache.
- Once the redo operation is completed, the DBR process updates all users' checkpoints to be the current AIJ end-of-file.

3. Performs the undo operation for each aborted user's incomplete transaction, if any. The DBR process reads the before-images from the user's .ruj file and writes them back to the database. If the dbkey also exists in a row cache, then the before-image is also written to its row cache entry.

4.4.3 RCS Process and Database Recovery

Because the RCS process and the DBR process both access the row cache structures, they must coordinate their activities. When a DBR process is activated, it immediately notifies the RCS process of its existence by using a lock. Then the RCS process aborts whatever request it is performing, requeues the request at the head of the appropriate queue, and waits for the database recovery activity to complete. Upon completion of database recovery, the RCS process resumes its operations by executing the next operation based on priority.

4.5 Considerations When Using the Row Cache Feature

This section contains further information on using the row cache feature.

- Hot standby

  Row caching is not allowed to be active on the standby database. Because the .aij file does not contain logical dbkeys, there is no way to maintain rows in the cache on the standby system. On the standby system, issue the RMU Open command with the Row_Cache=Disabled qualifier to open the database without activating row caching. If failover is necessary, close the standby database and reopen it normally. The standby database will have row caches activated.

- Backing files

If you are using row cache backing files, then do not use hot standby on the same machine as the master database. Both databases will attempt to use the same backing files.

Similarly, do not attempt to use the same directory location for backing files for two or more databases if any of their row cache names are identical. Multiple databases will attempt to use the same backing files.

- Utilities that access the database pages directly

  Some RMU commands do not access data by logical dbkey but instead read the database pages directly. These commands cannot access the row caches directly. Oracle Rdb resolves this problem by having each command request that the RCS process write all marked rows back to the database. The RMU operation waits for this task to complete.

  The RMU commands affected by this are:

  – Backup online

  – Analyze

  – Verify

  – Copy database online

  These operations may exhibit a delay in starting. If you specify the RMU log qualifier, Oracle Rdb will output a message when it is waiting for the RCS request and when the RCS request has completed. If your database row caches are set to checkpoint to the database rather than to backing files, then this delay will be minimized.

- Sequential scans

  When the execution strategy for a query is a sequential scan, Oracle Rdb scans the physical areas by performing the same I/O operations it would do if there were not any row caches. The major reasons for this are as follows:

  – Oracle Rdb does not have a list of all dbkeys in an area; it creates the list by reading all pages and examining all lines on each page. However, data is returned from the row cache if it is found there. Although Oracle Rdb reads the database pages to find the dbkeys of rows in the table, it still needs to look in the cache to see if the row is there. A row in the cache contains more recent data than that which is on disk.

  – There is no guarantee that all rows in a sequential scan can fit in a row cache. Row caches are often sized to include a percentage of the total number of rows where the most commonly used rows can be shared in memory.

Oracle Rdb is designed to avoid populating the cache during a strict sequential scan. It is designed this way because otherwise a query performing a sequential scan of a table looking for just a few records would fill the cache with every record and might force existing data in the cache back to disk. This would result in a row cache filled with records that you do not need in the cache.

However, note that a sequential *index* scan will populate the cache with data, index rows, or both.

- Snapshots enabled

  The Oracle Rdb snapshot mechanism of preserving a consistent view of the database for read-only transactions is not changed by the row cache feature. The before-images of rows needed by read-only transactions are preserved when read/write transactions write them to the snapshot files. Therefore, when snapshots are enabled, update operations are written to the rows in the row cache and the before-image of the row is written to disk. Oracle Rdb has optimized the snapshot mechanism with row caches, however, so that the performance of readers and writers may be better with row caches than without.

  The performance of row caches is typically much faster when snapshots are disabled. All of the disk I/O operations necessary to read and write to the snapshot file are eliminated. This is the ideal situation.

- Fragmented rows

  Fragmented rows are not stored in the row cache. They are created by fetching the fragments from the database and materializing them in process-private virtual memory.

- Vertical record partitioning

  When a logical cache is defined for a vertically partitioned table, each partition of a row is cached as a separate row cache entry. Only partitions that your query references and that can fit are inserted into the row cache.

- Unexpected storage area growth

  Oracle Rdb has optimized row caching to minimize the disk I/O operations required. Frequently operations are performed in memory only. Having the faster performance of in-memory updates is beneficial. However, when you make modifications that keep a row at its current size or smaller, or you make deletions, the database page does not reflect the amount of space that is in use.

Even though the row is logically smaller or erased from the database, it has not been physically removed from the database page. The space it occupies cannot be reused by another transaction until this row is finally written back to the database, usually by the RCS process during a sweep or checkpoint operation, depending on your row cache settings. Because of this, storage areas may grow larger than anticipated. If space reclamation is critical for some storage areas, then consider checkpointing their row caches to the database on a regular basis.

4.6 Requirements for Using Row Caches

To use the row cache feature, an Oracle Rdb database must meet the following configuration requirements:

- The number of cluster nodes must be one.
- After-image journaling must be enabled.
- Fast commit must be enabled.
- One or more row cache slots must be reserved.
- Row caching must be enabled.

Use the RMU Dump command with the Header qualifier to see if you have met the requirements for using row caches. In the following example, warnings are displayed for row cache requirements that have not been met.

```
$ RMU/DUMP/HEADER INVENTORY
    .
    .
    .
    Row Caches...
      - Active row cache count is 4
      - Reserved row cache count is 20
      - Checkpoint information
          Time interval is 10 seconds
          Default source is updated rows
          Default target is backing file
          Default backing file directory is "DISK1:[CACHE]"
      - WARNING: Maximum node count is 16 instead of 1
      - WARNING: After-image journaling is disabled
      - WARNING: Fast commit is disabled
     .
     .
     .
```

4.7 Designing and Creating a Row Cache

The following sections describe considerations for designing and creating row caches.

4.7.1 Reserving Slots for Row Caches

When you create a database, reserve enough row cache slots for both current and future needs. To reserve additional slots and to add or drop a row cache, the database must be closed.

Use the RESERVE n CACHE SLOTS clause of the CREATE DATABASE or ALTER DATABASE statement to reserve slots for row caches, as shown in the following example:

```
SQL> CREATE DATABASE FILENAME INVENTORY
    .
    .
    .
cont>  RESERVE 20 CACHE SLOTS;
```

If you do not specify a RESERVE n CACHE SLOTS clause, Oracle Rdb reserves one slot by default.

4.7.2 Row Cache Types

The two types of row caches are described in the following list:

- Physical area row cache

  You can create a general row cache that is shared by all row types that reside in one or more storage areas. This is the basic type of row cache, called a **physical area row cache**. Because physical area row caches are defined for a storage area, multiple storage areas can map to the same physical area row cache. A physical area row cache can contain all row types in a storage area. In addition, when a physical area row cache is defined, all rows of different sizes in the specified storage area are candidates for the row cache.

  See Section 4.7.2.1 for an example of how to assign a storage area to a row cache.

- Logical area row cache

You create a logical area row cache by creating a row cache that uses the same name as an existing table or index. A **logical area row cache** is associated with all partitions, both horizontal and vertical, of a specific table or index.

A logical area row cache cannot store the system row from a database page in a mixed format area.

You can use both physical and logical row caches to store a table and its index.

The following example shows the reason for using different row caches for different row types. Assume the following sizes for the rows in a table and hashed index:

- System records of 16 bytes
- Hash buckets of 100 bytes
- Data rows of 320 bytes

If you created one row cache for all three row types, with a row size of 320 bytes, much of the allocated memory would be wasted when storing the smaller system record and the hash bucket. Using this method, the amount of memory, excluding overhead, used for one row cache is as follows, assuming 15000 rows in the cache:

```
Total
number     = (# of rows in cache * row length of largest row)
of bytes

           = (15000 * 320)
           = 4800000 bytes
```

It is more efficient to have three caches, one for each of the row types:

- System records of 16 bytes (PARTS_SYS cache)
- Hash buckets of 100 bytes (PARTS_HASH cache)
- Data rows of 320 bytes (PARTS cache)

In this example, the system records are stored in a physical cache (PARTS_SYS) while the hash index buckets and data rows are stored in logical caches (PARTS_HASH and PARTS).

The amount of memory, excluding overhead, used with three row caches is computed as follows:

```
Total
number    = (# of rows in cache * row length of system record) +
of bytes    (# of rows in cache * row length of hash bucket) +
            (# of rows in cache * row length of data row)
```

```
= (5000 * 16) +
  (5000 * 100) +
  (5000 * 320)

= 2180000 bytes
```

4.7.2.1 Assigning Storage Areas to Row Caches

When a storage area is associated with a row cache, the row cache can contain all types of rows, if they fit. This is called a physical area row cache. One storage area can point to one row cache only. Multiple storage areas can be mapped to the same row cache.

You can also define a default row cache for all of the storage areas in the database by using one of the following statements:

- ALTER DATABASE ... ADD STORAGE AREA ... CACHE USING
- ALTER DATABASE ... ALTER STORAGE AREA ... CACHE USING
- CREATE DATABASE ... CREATE STORAGE AREA ... CACHE USING

The following example shows how to assign the same physical row cache to multiple storage areas:

```
SQL> ALTER STORAGE AREA
cont> PART_ID_A_E CACHE USING PARTS_SYS;
SQL> ALTER STORAGE AREA
cont> PART_ID_F_K CACHE USING PARTS_SYS;
```

4.7.2.2 Assigning Tables to Row Caches

A row cache is considered to be a logical area cache if its name is identical to the name of either a table or an index. If a logical area row cache is created for a vertically or horizontally partitioned table or horizontally partitioned index, then all rows in these partitions are mapped to the single logical area row cache.

In the following example, a logical area cache called PARTS is created for the PARTS table that is horizontally partitioned across five storage areas:

```
SQL> CREATE STORAGE MAP PARTS_MAP FOR PARTS
cont> --
cont> -- Parts table partitioned by part_id
cont> --
cont>  STORE USING (PART_ID)
cont>      IN PART_ID_A_E WITH LIMIT OF ('F')
cont>      IN PART_ID_F_K WITH LIMIT OF ('L')
```

```
cont>      IN PART_ID_L_P WITH LIMIT OF ('Q')
cont>      IN PART_ID_Q_U WITH LIMIT OF ('V')
cont>      OTHERWISE IN PART_ID_V_Z
cont>      PLACEMENT VIA INDEX PARTS_HASH;
SQL>
   .
   .
   .
SQL> ALTER DATABASE FILENAME INVENTORY
cont>     ADD CACHE PARTS
cont>         ROW LENGTH IS 100 BYTES
cont>         CACHE SIZE IS 5000 ROWS;
```

Rows from all five partitions of the PARTS table are automatically cached in the PARTS row cache, if they fit.

4.7.3 Sizing a Row Cache

When you size a row cache, you specify the following:

- Slot size

  The slot size is the fixed length size of each entry in the row cache. This determines the size of the largest row that can be stored in the row cache. Oracle Rdb will not cache a row if it is larger than the cache's slot size. Use the ROW LENGTH IS parameter of the ADD, ALTER, or CREATE CACHE clause to specify the slot size of the row cache.

  Oracle Rdb automatically rounds up the row length to the next 4-byte boundary. This is done because longword-aligned data structures perform optimally on its supported platforms.

  If you do not specify a slot size when creating a logical cache, Oracle Rdb generates a slot size based on the size of the table row or index node. Note, however, that Oracle Rdb finds the nominal row length of tables and indexes using the area inventory page (AIP). Under certain circumstances, this AIP length may not be the actual length of the row. In addition, some index structures may have no AIP entry at all. If no entry can be found, Oracle Rdb uses a default length of 256 bytes. Also, if the metadata for a table is modified, then the AIP length is not automatically updated. This can result in incorrect cache sizing. See the *Oracle Rdb7 Guide to Database Performance and Tuning* for more details on AIP lengths.

- Slot count

The slot count is the number of rows that can be stored in the cache. Use the CACHE SIZE IS parameter of the ADD, ALTER, or CREATE CACHE clause to specify the number of rows that can be stored in the cache.

If you do not specify the CACHE SIZE clause, Oracle Rdb creates a cache of 1000 rows by default.

The following example shows a row cache definition:

```
SQL> ADD CACHE PARTS
cont> ROW LENGTH IS 320 BYTES
cont> CACHE SIZE IS 3000 ROWS;
SQL> --
SQL> -- In this example, the slot size is 320 bytes
SQL> -- and the slot count is 3000.
SQL> --
```

It is important to select a proper slot size for the row cache. As stated previously, if a row is too large, Oracle Rdb will not cache the row. This can result in poor system performance because Oracle Rdb always checks the cache for the row before retrieving the row from disk. Use the RMU Dump Area command to determine the sizes of the data rows, hash buckets, and B-tree nodes. Keep in mind that row sizes within a table can vary greatly. If, for example, the largest row stored in a table is 100 bytes, but the majority of the rows range between 40 and 50 bytes, you may not want to choose 100 bytes for the slot size. However, you should account for most of the rows, including overhead. If you automatically select the largest row size without comparing it to the sizes of the other rows in the table, you might waste memory.

The following example dumps a few pages from the MY_AREA storage area:

```
$ RMU/DUMP/AREA=MY_AREA/START=5/END=10 TEST_DB/OUT=RMU_DUMP_AREA.OUT
```

Search the RMU_DUMP_AREA.OUT file for the occurrences of "total hash bucket" and "static data" as follows:

```
$ 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
  .
  .
  .
```

The hash bucket size is 118 bytes and the data row size is 311 bytes. Other rows in this table may require more or less space. It is important to scan a representative sample of random pages to determine the appropriate row size. Oracle Rdb rounds row sizes up to the next longword.

The RMU Show Statistics row caching screens provide performance information on inserting rows into a cache. One of the statistics, "row too big," indicates that a row is too large to fit into the specified cache. This statistic is also set when a row in a row cache becomes invalid and must be retrieved from the database page. For example, when a row in the row cache grows to the point where it becomes fragmented, it must be removed from the row cache. This is done by redirecting this row out of the row cache to disk, by setting its "row too big" attribute. See Section 4.10.2 for other information on the RMU Show Statistics screens related to row caching.

The slot count multiplied by the slot size specifies the approximate size, in bytes, of the row cache. You should also take into account additional overhead. See Section 4.7.4.1 for more information about sizing row caches.

4.7.4 Choosing Memory Location

When you create a row cache or modify a row cache definition, you have the option of specifying where in memory you want Oracle Rdb to create the cache. Row caches can reside in the following memory locations:

- Process global sections

  When you use global sections or shared memory created in the process space, you and other users share virtual memory, and the operating system maps a cache to a private address space for each user.

  Use the SHARED MEMORY IS PROCESS parameter to specify that the cache should be created in a process global section, shared memory partition, or shared memory, as shown in the following example:

  ```
  SQL> ALTER DATABASE FILENAME MF_PERSONNEL
  cont> ADD CACHE EMPIDS_LOW_RCACHE
  cont> SHARED MEMORY IS PROCESS;
  ```

  This is the default.

- System space buffer

  The system space global section is located in the OpenVMS Alpha system space, which means that a system space global section is fully resident, or pinned in memory, and does not affect the quotas of the working set of a process.

  System space is critical to the overall system. Because system space buffers are not paged, they use physical memory, thereby reducing the amount of physical memory available for other system tasks. This may be an issue if your system is constrained by memory. You should be careful when you allocate system space. Nonpaged dynamic pool (NPAGEDYN) and the VMScluster cache (VCC) are some examples of system parameters that use system space.

  Use the SHARED MEMORY IS SYSTEM parameter to specify that the cache be created in a system space buffer, as shown in the following example:

  ```
  SQL> ALTER DATABASE FILENAME MF_PERSONNEL
  cont> ADD CACHE EMPIDS_MID_RCACHE
  cont> SHARED MEMORY IS SYSTEM;
  ```

  Consider allocating small caches that contain heavily accessed data in system space buffers. When a row cache is stored in a system space buffer, there is no process overhead, and data access is very fast because the data does not need to be mapped to user windows. Also, OpenVMS Alpha systems (version 7 and later), make additional system space available by moving page tables and balance slots into very large memory (VLM) space. The Hot Row Information screen in the RMU Show Statistics command displays a list of the most frequently accessed rows for a specific row cache.

- Very large memory

  Very large memory (VLM) on OpenVMS Alpha systems allows Oracle Rdb to use as much physical memory as is available on the system and to dynamically map it to the virtual address space of database users. VLM provides access to a large amount of physical memory through small virtual address windows. Even though VLM is defined in physical memory, the virtual address windows are defined and maintained in each user's private virtual address space or system space, depending on the memory setting.

  Use the LARGE MEMORY parameter to specify that the cache be created in large memory.

```
SQL> ALTER DATABASE FILENAME MF_PERSONNEL
cont> ADD CACHE EMPIDS_OVER_RCACHE
cont> LARGE MEMORY IS ENABLED;
SQL>
```

  VLM is useful for large tables with high access rates. The only limiting factor with VLM is the amount of available physical memory on your system.

You view the physical memory through windows. You can specify the number of window panes with the WINDOW COUNT parameter. By default, Oracle Rdb allocates 100 window panes to a process.

Table 4–1 summarizes the location in memory of each row cache object and indicates whether process private virtual address windows are needed to access the data.

Table 4–1 Memory Locations of Row Cache Objects

SHARED	LARGE	Control Structures	Data Rows	Windows
PROCESS**[1]**	DISABLED**[3]**	Process global section or shared memory partition	Process global section or shared memory partition	No
PROCESS**[1]**	ENABLED**[4]**	Process global section or shared memory partition	Physical memory	Yes
SYSTEM**[2]**	DISABLED**[3]**	System space	System space	No
SYSTEM**[2]**	ENABLED**[4]**	System space	Physical memory	Yes

The following callouts describe the data presented in Table 4–1:

1. SHARED MEMORY IS PROCESS
   - The row cache control structures are located in a process global section or shared memory partition.
   - The storage of the data rows depends on whether large memory is enabled or disabled.
     * If large memory is enabled, data is stored in physical memory and windows from each user's process virtual address space are needed to access the data.
     * If large memory is disabled, data is stored in a process global section or shared memory partition and no windows are needed to access the data.
2. SHARED MEMORY IS SYSTEM
   - The row cache control structures are stored in system space.
   - The storage of the data rows depends on whether large memory is enabled or disabled.
     * If large memory is enabled, data is stored in physical memory and windows from each user's process virtual address space are needed to access the data.
     * If large memory is disabled, data is stored in system space and no windows are needed to access the data.
3. LARGE MEMORY IS DISABLED
   - The storage of the data rows and the row cache control structures depends on whether shared memory is process or system.
     * If shared memory is process, the data and row cache control structures are stored in a process global section or shared memory partition and no windows are needed to access the data.
     * If shared memory is system, the data and row cache control structures are stored in system space and no windows are needed to access the data.
4. LARGE MEMORY IS ENABLED
   - The data rows are stored in physical memory and process private virtual address windows are needed to access the data.

- The storage of the row cache control structures depends on whether shared memory is process or system.
  * If shared memory is process, the control structures are stored in a process global section or shared memory partition.
  * If shared memory is system, the control structures are stored in system space.

It is important to consider the amount of memory available on your system before you start creating and using row caches.

On OpenVMS systems, you can use the DCL command SHOW MEMORY/PHYSICAL to check the availability and usage of physical memory. This command displays information on how much memory is used and how much is free. The free memory is available for VLM row caches in addition to user applications.

Because VLM row caches can consume a certain amount of system space for their virtual address windows, Oracle Corporation recommends that you define the VLM row caches first, so that any VLM system space requirements are satisfied before you define system space buffer row caches for small tables that contain frequently accessed data.

The following example shows a system that has 1.5 gigabytes of memory or a total of 196608 OpenVMS Alpha memory pages (an OpenVMS Alpha page is 8192 bytes):

```
$ 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
```

Of the 1.5 gigabytes, 183605 pages remain on the free list. Most of this free memory is available for row cache allocation.

Assume that a logical area cache has been defined for the MY_TABLE table. The following SQL statement maps the logical area cache:

```
SQL> ATTACH 'FILE TEST_DB';
SQL> SELECT * FROM MY_TABLE WHERE MY_HASH_INDEX = 100;
```

By issuing this SQL statement, the logical area cache has allocated the necessary memory, accounting for 40462 OpenVMS Alpha pages, as shown in the following SHOW MEMORY/PHYSICAL command output. Subtract the free space in the following example from the free space in the previous example to find the pages used for the logical area row cache: `183605 - 143143 = 40462`

```
$ 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
```

Notice that the amount of free memory has been reduced.

The following SHOW MEMORY/PHYSICAL command was issued after users attached to the database, allocated their working sets, and began to work:

```
              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
```

In this example, only 81046 OpenVMS Alpha pages are left on the free list.

4.7.4.1 Sizing Considerations

The following information is intended to help you determine in which memory location to place your row cache based on system resources. Generally, if your row cache will fit into a process global section or system space buffer, then it will perform slightly better. If space is an issue, then you should place the row cache in VLM.

When a row cache is created in a process global section or system space buffer, Oracle Rdb sizes it using the following values:

- Each slot requires 48 bytes plus the length of the slot rounded to the next 4-byte boundary.
- Each row cache requires a hash table of (4 * (the number of cache slots rounded to the next higher power of 2)) bytes.
- Each row cache requires (24 * the maximum number of users) bytes.

When a row cache is created in VLM, Oracle Rdb sizes it using the following values:

- Each slot requires 24 bytes plus the length of the slot rounded up to the next 4-byte boundary.

When VLM is enabled and the cache is created in a process global section or system buffer space, Oracle Rdb sizes it using the following values:

- Each slot requires 24 bytes.
- Each cache requires a hash table of (4 * (the number of cache slots rounded up to the next higher power of 2)) bytes.
- Each cache requires (24 * the maximum number of users) bytes.

Example 4–1 shows how Oracle Rdb sizes a cache containing 150,000 slots with a slot size of 500 bytes in a process global section or system space buffer and a maximum of 350 users. (Note that 2 to the 17th power is 262144.)

Example 4–1 Sizing a Row Cache in a Global Section or System Space Buffer

```
Total
number  = (150000*(500+48)) + (262144*4) + (24*350)
of
bytes

        = 83,256,976 bytes
```

Example 4–2 shows how Oracle Rdb sizes the same cache in VLM.

Example 4–2 Sizing a Row Cache in VLM

```
Total
number  = (150000*(500+24))
of
bytes

        = 78,600,000 bytes
```

Example 4–3 shows how Oracle Rdb sizes the same cache in a process global section or system space buffer with VLM enabled.

Example 4–3 Sizing a Row Cache in Memory with VLM Enabled

```
Total
number  = (150000*24) + (262144*4) + (24*350)
of
bytes

        = 4,656,976 bytes
```

4.8 Using the Row Cache Feature

The following sections describe how to set parameters for the row cache feature.

4.8.1 Enabling and Disabling Row Cache

Row caching can be enabled or disabled in three ways:

1. You can enable row caching for a database by using the ROW CACHE IS ENABLED clause of the SQL ALTER DATABASE or CREATE DATABASE statement.

   The following example shows how to enable the row cache feature and its requirements:

   ```
   SQL> ALTER DATABASE FILENAME MF_PERSONNEL
   cont> NUMBER OF CLUSTER NODES IS 1
   cont> JOURNAL ENABLED (FAST COMMIT ENABLED)
   cont> RESERVE 20 CACHE SLOTS
   cont> ROW CACHE IS ENABLED;
   ```

   You can disable row caching for a database by using the ROW CACHE IS DISABLED clause of the SQL ALTER DATABASE or CREATE DATABASE statement:

   ```
   SQL> ALTER DATABASE FILENAME MF_PERSONNEL
   cont> ROW CACHE IS DISABLED;
   ```

   Row caching is also disabled if one of the conditions described in Section 4.6 becomes false.

   When row caching is disabled, all previously created and assigned row caches remain in existence for future use when row caching is enabled again.

   The database must be closed when you enable or disable row caching.

2. The RMU Set command allows you to enable or disable row caching using an unjournaled operation. This is needed to disable row caches if you have system tables mapped to row caches and you need to perform SQL operations that require exclusive database access.

   ```
   RMU/SET/ROW_CACHE[/DISABLED|/ENABLED] database_name
   ```

   For example, adding a row cache to a database requires exclusive database access. Execute this command before adding a new row cache using SQL, then enable row caching again.

3. The RMU Open command with the Row_Cache=Disabled qualifier is used to keep row cache enabled in the database but not used for the duration of the open. This is necessary to set up row caching in a hot standby environment. Row caching is not allowed to be active on the standby database. Therefore, issue this command on the standby system to open the database without activating row caching.

4.8.2 Specifying Checkpointing and Sweeping Options

The following sections provide guidelines for specifying checkpointing and sweeping options.

4.8.2.1 Choosing the Checkpoint Source and Target Options

For greatest flexibility, provide each row cache with its own checkpoint source and target options as follows:

- The source rows to read

  This determines which source rows in the cache to write back to disk. Only updated rows or all rows can be selected. By default, only updated rows are selected.

- The target location to write the rows

  This determines whether the source rows are written back to the database pages or written out to a separate row cache backing file.

You can specify the target location using the following parameters of the ADD, ALTER, and CREATE CACHE clauses. Note that you cannot specify that all rows are checkpointed to the database.

- CHECKPOINT UPDATED ROWS TO BACKING FILE
- CHECKPOINT UPDATED ROWS TO DATABASE
- CHECKPOINT ALL ROWS TO BACKING FILE

Table 4–2 lists the advantages and disadvantages of each checkpoint target.

Table 4–2 Checkpoint Target Options

Advantages	Disadvantages
Checkpoint to Database	
Does not require any more disk space.	Is slower due to contention for database page buffers.

Table 4–2 (Cont.) Checkpoint Target Options

Advantages	Disadvantages
Checkpoint to Database	
Simpler to understand because it uses the traditional database page buffers.	Upon node failure, the row cache is not repopulated.
Unmarks slots in the row cache so they can be reused for other rows.	Greater conflict with other users because row and page locks are maintained. The row cache server (RCS) process does not respond to requests to release row or page locks.
Writing back to database pages reclaims space on database pages from erased or modified rows that have been reduced in size.	
Checkpoint to Backing File	
Can checkpoint all rows allowing a way to repopulate row caches that are predominantly read-only while recovering from a node failure.	Requires extra disk space to create two backing files per cache.
Faster at writing sequential I/O operations to backing file.	Used only for node failure protection.
Can be placed on different spindles so that other database I/O activity will not be impacted.	Marked rows tend to stay marked. By definition, rows in a row cache are unmarked only when they are written back to the database.
Used upon node failure to repopulate the row cache.	Space on the database pages resulting from erased rows and modified rows that are reduced in size is not reclaimed.

4.8.2.2 Choosing the Checkpoint Interval

You must specify a checkpoint interval by using the CHECKPOINT TIMED EVERY s SECONDS parameter of the ROW CACHE IS ENABLED clause. This checkpoint parameter applies to the RCS process only.

This value can be overridden by the RDM$BIND_CKPT_TIME logical name (this logical name is also used to override the FAST COMMIT checkpoint interval). If nothing is specified, Oracle Rdb uses a default checkpoint interval of 15 minutes.

4.8.2.3 Specifying Sweeping Parameters

You set the number of updated rows that will be swept by using the NUMBER OF SWEEP ROWS IS parameter of the ADD, ALTER, or CREATE CACHE clause.

```
SQL> ALTER DATABASE FILENAME INVENTORY
cont> ALTER CACHE PARTS
cont> ROW LENGTH IS 104 BYTES
cont> CACHE SIZE IS 2000 ROWS
cont> CHECKPOINT ALL ROWS TO BACKING FILE
cont> NUMBER OF SWEEP ROWS IS 200;
```

A row in a row cache cannot be reused if it is marked (modified) or if its reference count is greater than zero. In the latter case, one or more users have a reference to this row in their row cache working sets. The RCS sweep operation tries to eliminate these restrictions from rows in the row cache so these rows can be reused to insert new rows.

The RCS process writes committed modified rows back to the database, up to a maximum of the NUMBER OF SWEEP ROWS defined for the row cache. It is important that this value be set properly so that when a sweep is initiated, the RCS process clears out enough slots to allow sufficient insertion activity before another sweep operation is necessary. Typically, a value of 10 percent to 30 percent of the size of the row cache would be sufficient. Make sure that the sweep count is larger than the value of the row cache's reserved count, specified by the NUMBER OF RESERVED ROWS IS N clause.

You can override the row cache's defined sweep count value by defining the RDM$BIND_RCS_SWEEP_COUNT logical name. Note, however, that the value of this logical name applies to all row caches.

During a sweep operation, the RCS process may also initiate a dialogue with current users to reset the reference counts of the rows in the cache. The RCS process will only do this during a sweep operation if the number of database recovery processes since the last sweep operation of this row cache has exceeded the number specified by the RDM$BIND_RCS_CLEAR_GRICS_DBR_CNT logical name. Only processes that have abnormally terminated fail to clean up their reference counts normally.

An RCS sweep operation is triggered when a row cache is considered *clogged*. A row cache is considered clogged when a user fails to find any available slots in which to insert rows. Even after a row cache is considered full, a user may still be able to insert rows into that row cache if the user still has reserved slots to use.

The RCS process clears the clogged flag if the sweep operation was successful in opening up some slots. The clogged flag can also become clear during a checkpoint operation if the RCS process has detected row cache entries with zero reference counts. This happens only if the clogged flag stays set for three consecutive checkpoint operations.

4.8.2.4 Specifying the Size and Location of the Cache Backing File

When allocating the size of the cache backing (.rdc) files, consider the following:

- Whether all rows or only marked rows will be checkpointed
- The amount of update activity in the row cache
- Whether you want to create new backing files on each database open or reuse existing backing files

If you want Oracle Rdb to automatically rebuild an entire row cache in memory after a node failure, then define the row cache to checkpoint all rows to a cache backing file. If you want Oracle Rdb to repopulate the row cache with only the rows that were modified at the time, then define the row cache to checkpoint only updated rows to the cache backing file.

The decision that you make determines how to size the cache backing files.

If all rows are to be checkpointed, use the following formula to determine the number of blocks to allocate for the cache backing file.

```
Number of
blocks    = (slot count * (row length + 40)) / 512 bytes per block
```

If only the updated rows are to be written to the backing file, use the following formula to allocate the backing file, based on the estimated number of updated rows in the row cache.

```
Number of
blocks    = (# of updated rows * (row length + 40)) / 512 bytes per block
```

You can overwrite the allocation specified in the row cache definition with the RDM$BIND_CKPT_FILE_SIZE system logical name. This specifies the percentage of the row cache size to allocate for the backing file. The default is 40 percent.

```
Number of
blocks    = (0.40 * slot count * (row length + 40)) / 512 bytes per block
```

When checkpointing to backing files, Oracle Rdb needs two backing files for each cache. One is used for the last checkpoint (committed rows), and the other is used for the current checkpoint. Make sure there is enough disk space for two backing files for each cache. By default, Oracle Rdb deletes the backing files upon successful database shutdown and re-creates them when the database is reopened. If you prefer, you can tell Oracle Rdb to save the backing files and reuse them on the subsequent database open by defining the system logical name RDM$BIND_RCS_KEEP_BACKING_FILES to 1.

If you are checkpointing a row cache to the database, you do not need to specify an allocation or location for the cache backing file. Oracle Rdb will ignore these clauses.

If you have a read-only cache, specify 1 block for the size of the cache backing file as follows:

```
SQL> ALTER DATABASE FILENAME MF_PERSONNEL
cont> ADD CACHE RCACHE_2
cont> LOCATION IS WORK$DISK1:[RCS]
cont> ALLOCATION IS 1 BLOCK;
```

4.9 Controlling What Is Cached in Memory

The ROW REPLACEMENT parameter of the ADD, ALTER, and CREATE CACHE clause gives you some control over what happens when a row cache becomes full. If row replacement is enabled for a particular row cache, new rows will replace the oldest, unused, unmarked rows once the cache is full. If row replacement is disabled, new rows are not placed in the cache once the cache is full; they will always be retrieved from disk.

When you use the ROW REPLACEMENT IS DISABLED parameter, the data that was memory resident stays that way. Therefore, all subsequent reads of that data occur from memory rather than disk.

You can increase performance by making the following types of rows memory resident.

- Nonleaf nodes of a B-tree index

  Be sure to account for the nodes splitting when you specify the size for the row cache. If a parent node splits and there is no room in the cache for the new node, the new node will not be held in memory.

- Data that is primarily read-only

  Data that does not change very often, such as dimension tables in a data warehouse environment, is a good candidate for keeping resident in memory.

- Data that is update-intensive (if the entire table can fit in the cache)

  Oracle Rdb optimizes access when the cache is defined with row replacement disabled.

Enabling row replacement is beneficial when access patterns of a table are random. This ensures that the most frequently accessed rows remain in memory. Often, there may not be enough physical memory to cache an entire table, so caching the most frequently used rows can improve performance.

4.9.1 Row Replacement Strategy

Global and local buffers use the least recently used (LRU) replacement strategy for database pages. Row caching uses a modified form of the LRU replacement strategy. Each database user can protect the last 10 rows that he or she accessed. This group of rows is referred to as a **working set**. Rows that belong to a working set are considered to be **referenced** and are not eligible for row replacement. Any row that is in a cache and is not part of a working set is considered to be an **unreferenced** row. The unreferenced rows are eligible for replacement if they are not marked.

4.9.2 Inserting Rows into a Cache

Each user process requests rows from the database. A user process, which reads a row from a storage area, tries to insert the row into the cache (if it is not already there). If a slot is available, the requested row is stored in the cache, if it fits. If no more slots are available in the cache, one of the following happens:

- If ROW REPLACEMENT IS ENABLED and an unmarked, unreferenced row can be found, that row is replaced by the new row. Oracle Rdb chooses the unreferenced row randomly.
- If ROW REPLACEMENT IS DISABLED, then the row is not stored in the cache. This means that when the cache fills, it will not accept new rows. Reserved slots, however, can still be used.

You can prevent individual processes from inserting new rows into any Oracle Rdb row cache by defining the process logical name RDM$BIND_RCACHE_INSERT_ENABLED to 0. When defined, a process can only use what already exists in the row caches; the process cannot insert a row into a row cache. This option is useful if, for example, you want to keep nightly batch processes that perform large reporting functions from filling up row caches that are also used by the more important, daily, online transaction processing servers.

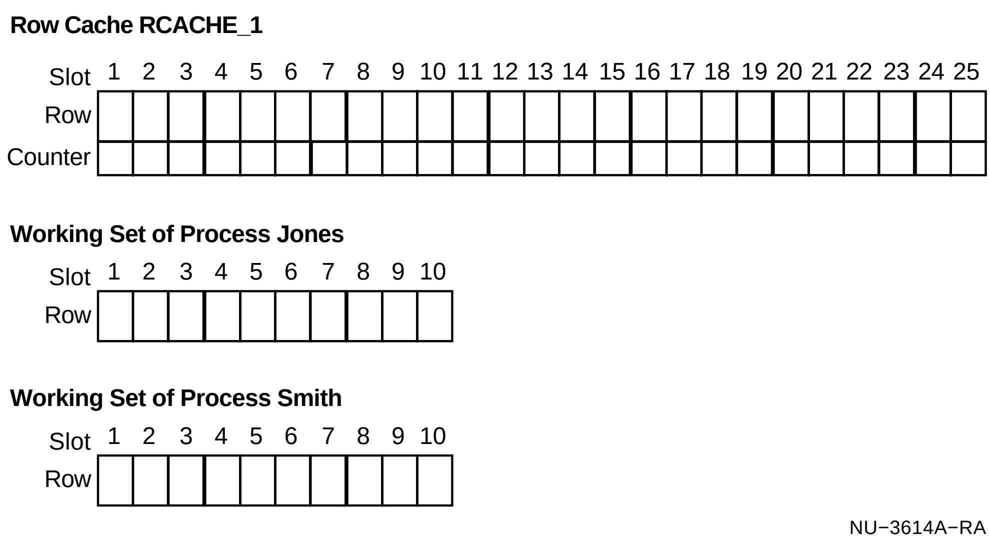
If system usage is lighter at night, you may want to preload row caches so that the data is available in memory during the day when database activity is at its peak.

The remainder of this section illustrates how Oracle Rdb inserts rows into a cache.

The example makes the following assumptions:

- Row caching is enabled.
- Row replacement is enabled.
- A row cache (RCACHE_1) has been created with 25 slots.
- Two processes (Jones and Smith) are attached to the database.
- The rows in the row cache are not modified.

The initial allocation is as follows:



1. Process Jones executes a query that causes 5 rows to be read into the first 5 slots of the row cache.

Slot	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
Row	A	B	C	D	E																				
Counter	1	1	1	1	1																				

Working Set of Process Jones

Slot	1	2	3	4	5	6	7	8	9	10
Row	A	B	C	D	E					

NU−3615A−RA

Each row slot has a working set counter associated with it. The working set counter indicates whether the row belongs to a working set. A positive value indicates that the row belongs to a working set. If a row belongs to a working set, it is not eligible for row replacement.

2. Process Smith requests 15 rows from the database. The first 10 rows requested go into Smith's working set as follows:

Working Set of Process Smith

Slot	1	2	3	4	5	6	7	8	9	10
Row	F	G	H	I	J	K	L	M	N	O

NU−3616A−RA

Process Smith's working set has exactly 10 slots, and all 10 are being used. The least recently used row is replaced by the eleventh row that process Smith reads into the cache. Rows 12 through 15 also overwrite the contents of slots 2 through 5 respectively.

After the 15 rows are read into the cache, the cache appears as follows:

Slot	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
Row	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T					
Counter	1	1	1	1	1	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1					

NU−3617A−RA

After the 15 rows are read into the cache, process Smith's working set appears as follows:

Working Set of Process Smith

Slot	1	2	3	4	5	6	7	8	9	10
Row	P	Q	R	S	T	K	L	M	N	O

NU−3618A−RA

At this point, rows F, G, H, I, and J are unreferenced. They are in the cache but they do not belong to the working set of any process. Oracle Rdb sets the working set counter for an unreferenced row to zero. The unreferenced rows are eligible for replacement if they have not been modified and row replacement is enabled. Any process can read rows F, G, H, I, or J without executing an I/O operation. However, if a process requires a row that is not currently in the cache, one of the rows F, G, H, I, or J is replaced with the new row.

Each slot in the row cache contains a modification flag. If the row has been modified but not yet flushed to disk, it is considered to be **dirty**. Dirty rows are not candidates for row replacement either. Modified rows are written to disk by the row cache server (RCS) process. See Section 4.8.2.1 for more information.

3. Process Jones requests 7 more rows: M, U, V, W, X, Y, and Z. Jones can read row M without performing any I/O because M is already in the cache. An additional slot does not get filled in the row cache, but row M is added to process Jones' working set.

   Process Jones' working set now appears as follows:

Working Set of Process Jones

Slot	1	2	3	4	5	6	7	8	9	10
Row	Y	B	C	D	E	M	U	V	W	X

NU−3619A−RA

Rows U, V, W, X, and Y go into the remaining slots in the row cache and the row cache appears as follows:

Slot	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
Row	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y
Counter	0	1	1	1	1	0	0	0	0	0	1	1	2	1	1	1	1	1	1	1	1	1	1	1	1

NU−3620A−RA

Note that the working set counter for slot 13 indicates that row M is in two working sets. This indicates that two processes are accessing the same row. The number of processes sharing a particular slot is known as the **share count**.

At this point, the cache is full. If row replacement were disabled for the row cache, then row Z could not be inserted. However, in this example, row replacement is enabled, and there is an unreferenced slot. Therefore, Oracle Rdb will choose an unreferenced slot to make room for the new row, Z. (In this example, the unreferenced slots are A, F, G, H, I, and J.)

4.10 Examining Row Cache Information

There are two ways to display information about row caches. You can display the attributes using the SHOW CACHE statement or you can display statistics using the RMU Show Statistics command. The following two sections present these two methods.

4.10.1 Displaying Row Cache Attributes

You can use the SHOW CACHE statement to display attributes, as shown in the following example:

```
SQL> SHOW CACHE PARTS;
     PARTS
           Cache Size:         204 rows
           Row Length:         104 bytes
           Row Replacement:    Enabled
           Shared Memory:      Process
           Large Memory:       Disabled
           Window Count:       100
           Reserved Rows:      20
           Sweep Rows:         1004
           Allocation:         100 blocks
           Extent:             100 blocks
```

You can also use the RMU Dump command with the Header qualifier to display row cache information, as in Example 4–4.

Example 4–4 Displaying Row Cache Parameters

```
$ RMU/DUMP/HEADER INVENTORY
    .
    .
    .
    Row Caches...   [1]
      - Active row cache count is 4
      - Reserved row cache count is 20
      - Checkpoint information
          Time interval is 10 seconds
          Default source is updated rows
          Default target is backing file
          Default backing file directory is "DISK1:[RDB]"
    .
    .
    .
Row cache "PARTS"
    Cache ID number is 4 [2]
    Allocation...   [3]
      - Row slot count is 204
      - Maximum row size allowed in cache is 104 bytes
      - Working set count is 10
      - Maximum slot reservation count is 20
      - Row replacement is enabled
    Sweeping...   [4]
      - Sweep row count is 1004
      - Maximum batch I/O count is 0
    Checkpointing...  [5]
      - Source is updated rows (database default)
      - Target is backing file (database default)
      - No checkpoint information available
      - Checkpoint sequence is 0
    Files...   [6]
      - Default cache file directory is "DISK1:[RDB]"
      - File allocation is 100 blocks
      - File extension is 100 blocks
    Hashing...   [7]
      - Hash value for logical area DBIDs is 211
      - Hash value for page numbers is 11
    Shared Memory...  [8]
      - System space memory is disabled
```

```
  - Large memory is disabled
  - Large memory window count is 100
Cache-size in different sections of memory... [9]
  - Without VLM, process or system memory requirement
    is 309760 bytes
  - With VLM enabled...
      - Process or system memory requirement is 38768 bytes
      - Physical memory requirement is 280000 bytes
      - VLM Virtual memory address space requirement is
approximately 102400 bytes
```

The following callouts identify the parameters in Example 4–4.

1. Row Caches . . .
   - Active row cache count is 4

     This specifies the number of row caches currently defined in this database.
   - Reserved row cache count is 20

     This specifies the number of slots that are available in the database. The row cache slots are reserved with the RESERVE n CACHE SLOTS parameter of the ALTER or CREATE DATABASE statements.
   - Checkpoint information

     This displays database-level checkpoint information specified using parameters of the ADD, ALTER, or CREATE CACHE clauses.

     * Time interval is 10 seconds

       A checkpoint is one full pass through all active row caches, attempting to write all or just marked rows back to their respective storage areas or to the backing file. The time interval is set with the CHECKPOINT TIMED EVERY s SECONDS parameter.
     * Default source is updated rows

       Only updated rows are written to the backing file or back to the database storage areas.
     * Default target is backing file

       Specifies that the default target for the checkpoint is the backing file and not the database. This is the default target when the CHECKPOINT UPDATED ROWS parameter is not set.
     * Default backing file directory is "DISK1:[RDB]".

The default cache file directory is the directory where Oracle Rdb places the cache backing store files. If you do not explicitly include a directory specification, Oracle Rdb will place the backing file in the directory where the database root file is stored.

2. Cache ID number is

   Oracle Rdb assigns an ID to each defined row cache in the database.

3. Allocation . . .

   - Row slot count is 204

     This is specified with the CACHE SIZE IS n ROWS parameter.

   - Maximum row size allowed in cache is 104 bytes

     This is specified with the ROW LENGTH IS n BYTES parameter.

   - Working set count is 10

     This is the number of "in use" rows that are not eligible for row replacement.

   - Maximum slot reservation count is 20

     This is specified with the NUMBER OF RESERVED ROWS parameter. The default value is 20 rows.

     The number of reserved rows indicates how many slots in the cache Oracle Rdb will reserve for each process. Reserving many rows minimizes row cache locking while rows are inserted into the cache.

     The number of reserved rows parameter is also used when searching for available slots in a row cache. The entire row cache is not searched on the initial pass. This parameter is used as the maximum number of rows that are searched for a free slot. If at least one free slot is found, the insert operation can proceed. If no free slots are found in this initial search, Oracle Rdb will continue searching the cache until it finds a free slot.

   - Row replacement is enabled

     This is specified with the ROW REPLACEMENT parameter. Row replacement is enabled by default.

4. Sweeping . . .

   - Sweep row count is

Sets the number of marked rows that will be swept back to the database or backing file when the row cache is full and a user attempts to find an empty slot.

5. Checkpointing . . .

   - Source is updated rows (database default)

     The source of updated rows is the same as the database default.

   - Target is backing file (database default)

     The target for marked rows is the database default.

6. Files . . .

   - Default cache file directory is "DISK1:[RDB]"

     The LOCATION parameter specifies a directory specification for the cache backing store file. Oracle Rdb writes to the cache backing store file when the RCS process checkpoints. Oracle Rdb automatically generates a file name with a file extension of .rdc. The default location for the cache backing store file is the directory where the database root file is stored.

     The LOCATION parameter can be specified at the database level or at the row cache level. If you include the LOCATION parameter in the ADD CACHE or CREATE CACHE clause of the CREATE or ALTER DATABASE statement, the directory that you specify becomes the default directory location for all row caches that are defined for the database. You can, however, override the default directory location for individual row caches by specifying the LOCATION parameter in the row cache definition.

   - File allocation is 100 blocks

     The ALLOCATION parameter specifies the initial size of the cache backing file. The default allocation is 40 percent of the cache size. The cache size is determined by multiplying the number of rows in the cache by the row length.

   - File extension is 100 blocks

     The EXTENT parameter specifies the number of pages by which the cache backing store file can be extended after the initial allocation has been reached. The default extent is 127 multiplied by the number of rows in the cache.

7. Hashing . . .

   - Hash value for logical area DBIDs is 211

- Hash value for page numbers is 11

  The hash values are used by Oracle Rdb to fine-tune the distribution of hash table queues in the row cache.

8. Shared Memory . . .

   - System space memory is disabled

     This is specified with the SHARED MEMORY parameter. This specifies whether Oracle Rdb creates the row cache in shared memory. The row cache is created in a process global section by default.

   - Large memory is disabled

     This is specified with the LARGE MEMORY parameter. This specifies whether Oracle Rdb creates the row cache in physical memory. Large memory is disabled by default.

   - Large memory window count is 100

     This is specified with the WINDOW COUNT parameter. The default value is 100 windows. The WINDOW COUNT parameter specifies how many locations of the physical memory are mapped to each user's private window in virtual address space.

9. Cache-size in different sections of memory . . .

   - Without VLM, process or system memory requirement is 309760 bytes

     When the cache is created in a process global section or system space buffer and VLM is not enabled, this is the memory requirement.

   - With VLM enabled . . .

     * Process or system memory requirement is 38768 bytes

       When VLM is enabled and the cache is created in a process global section or system space buffer, this is the memory requirement.

     * Physical memory requirement is 280000 bytes

       The actual cached data requires this space in VLM.

     * VLM Virtual memory address space requirement is approximately 102400 bytes

       This is the address space used by the virtual memory windows.

4.10.2 RMU Show Statistics Screens and Row Caching

The RMU Show Statistics command displays information regarding row caches. The following are titles of some of the screens that display row cache information:

- 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

4.11 Examples

This section includes some practical examples on using the row cache feature of Oracle Rdb.

4.11.1 Loading a Logical Area Cache

Use the following steps to place an entire table in a row cache:

1. Determine how many rows are in the table.

```
SQL> SELECT COUNT(*) FROM EMPLOYEES;
         100
1 row selected
```

2. Create a logical cache large enough to hold the table.

   Use the table name as the name of the cache to create the logical cache. Oracle Rdb will determine the row length from the table.

```
SQL> ALTER DATABASE FILENAME MF_PERSONNEL
cont> ADD CACHE EMPLOYEES
cont> CACHE SIZE IS 100 ROWS;
```

3. Cause Oracle Rdb to sort the table by an indexed field.

   This causes rows to be read by DBKEY after the sort is complete.

```
SQL> SELECT * FROM EMPLOYEES ORDER BY EMPLOYEE_ID;
 EMPLOYEE_ID   LAST_NAME        FIRST_NAME   MIDDLE_INITIAL
   ADDRESS_DATA_1              ADDRESS_DATA_2         CITY
      STATE   POSTAL_CODE   SEX    BIRTHDAY      STATUS_CODE
  00197         Danzig           Chris        NULL
    136 Beaver Brook Circle                          Acworth
       NH      03601         F      21-Jun-1939   1
  .
  .
  .
```

4.11.2 Caching Database Metadata

Because metadata is frequently accessed, you may want to cache some or all of your database's metadata. You can map the entire contents of the RDB$SYSTEM storage area to a physical area row cache. Alternatively, you can map certain system tables, such as RDB$RELATIONS and RDB$INDICES, into separate logical area row caches.

To do this, follow these steps:

1. Use the RMU Dump command with the Area qualifier to display the contents of the storage area. (Note that the RMU Dump command output uses the term *records* to refer to rows.)

```
$ RMU/DUMP/AREA=RDB$SYSTEM/OUT=RMU_DUMP_1.OUT MF_PERSONNEL
$ SEARCH/STATISTICS RMU_DUMP_1.OUT "RECORD LENGTH", "STATIC_DATA"

                            00A2  0050  record length 162 bytes
                            00E8  008B  record length 232 bytes
                            00C4  00C6  record length 196 bytes
                            00E4  0101  record length 228 bytes
                            0088  013C  record length 136 bytes
                            023C  0177  record length 572 bytes
                            0220  01B2  record length 544 bytes
                            030C  01ED  record length 780 bytes
  .
  .
```

```
  .
Files searched:                1      Buffered I/O count:       100
Records searched:          62260      Direct I/O count:         441
Characters searched:     3459752      Page faults:               20
Records matched:              96      Elapsed CPU time:  0 00:00:01.63
Lines printed:                96      Elapsed time:      0 00:00:02.83
```

2. Determine the row length and slot count.

   Keep in mind that other structures may be stored in this area because it can be specified as the default storage area for Oracle Rdb.

3. Add the physical cache and assign it to the RDB$SYSTEM storage area.

   In the following example, row length has been rounded up and the row cache size has been increased to allow for future growth.

```
SQL> ALTER DATABASE FILENAME MF_PERSONNEL
cont> ADD CACHE RDB_SYSTEM_CACHE
cont> CACHE SIZE IS 9000 ROWS
cont> ROW LENGTH IS 800 BYTES;
SQL> ALTER DATABASE FILENAME MF_PERSONNEL
cont> ALTER STORAGE AREA RDB$SYSTEM
cont> CACHE USING RDB_SYSTEM_CACHE;
```

4. Or, add the logical area caches to the Oracle Rdb system tables of interest.

```
SQL> ALTER DATABASE FILENAME MF_PERSONNEL
cont>     ADD CACHE RDB$RELATIONS
cont>         CACHE SIZE IS 1000 ROWS
cont>         ROW LENGTH IS 500 BYTES
cont>     ADD CACHE RDB$INDICES
cont>         CACHE SIZE IS 2000 ROWS
cont>         ROW LENGTH IS 500 BYTES;
```

When caching metadata, you will experience conflicts when executing database operations through SQL that require exclusive database access. For example, adding new row caches or dropping existing ones requires exclusive database access. When the SQL command is parsed, the Oracle Rdb system tables are queried. This access to the system tables creates the row caches and causes the RCS process to come up to manage those row caches. As a result, the database now has another "user," the RCS process. This causes the exclusive database operation to fail.

To resolve this, you must first turn off row caching temporarily using the RMU Set command specifying the Row_Cache and Disabled qualifiers. Then, perform the

SQL operation that requires exclusive database access. Finally, reenable row caching using the RMU Set command with the Row_Cache and Enabled qualifiers.

4.11.3 Caching a Sorted Index

To cache a sorted index, use the following steps:

1. Display the number of index nodes using the RMU Analyze Index command. (Note that the RMU Analyze command uses the term *records* to refer to rows.)

   ```
   $ RMU/ANALYZE/INDEX MF_PERSONNEL EMP_LAST_NAME

    Index EMP_LAST_NAME for relation EMPLOYEES duplicates allowed
     Max Level: 2, Nodes: 8, Used/Avail: 1625/3184 (51%), Keys: 90, Records: 67
          Duplicate nodes: 16, Used/Avail: 264/312 (85%), Keys: 16, Records: 33
   ```

2. Count the number of nodes and duplicate nodes.
3. Allocate slots based on the number of nodes currently used and allow for future growth.

   In this example, allocating 28 slots would be reasonable.
4. Determine node and duplicate node size. Sorted indexes with duplicates should be sized at 430 bytes rounded up to the next 4-byte interval.
5. Create a logical cache for the sorted index.

   ```
   SQL> ALTER DATABASE FILENAME MF_PERSONNEL
   cont> ADD CACHE EMP_LAST_NAME
   cont> ROW LENGTH IS 440 BYTES
   cont> CACHE SIZE IS 28 ROWS;
   ```

4.12 Logical Names Relating to the Row Cache Feature

Table 4–3 describes logical names specific to row caching and explains when and how to use them.

Table 4–3 Logical Names Specific to Row Caching

Logical Name	Table in Which Logical Is Stored	Value	Description
RDM$BIND_CKPT_FILE_SIZE	LNM$FILE_DEV	INTEGER	Specifies the percentage of the row cache size that you want the backing file allocation to be. This is applied to all backing files. This overrides the backing file allocation specified in the CREATE or ADD CACHE definition.
RDM$BIND_CKPT_TIME	LNM$FILE_DEV	INTEGER	Specifies the frequency with which the RCS checkpoints. It overrides the value set with the SQL ALTER DATABASE ROW CACHE IS ENABLED (CHECKPOINT TIMED EVERY N SECONDS) command.
RDM$BIND_DBR_UPDATE_RCACHE	LNM$SYSTEM_TABLE	0 or 1	Specifies whether row caches in memory are repopulated after a node failure. If the logical name is set to 0 during recovery from a node failure, then row caches in memory are not repopulated from the backing files (only the database is recovered). If the logical name is set to 1 during a recovery from a node failure, then the row caches in memory are repopulated from backing files and from REDO operations. The default is 1.
RDM$BIND_RCACHE_INSERT_ENABLED	LNM$FILE_DEV	0 or 1	Specifies whether a process can insert new rows into the row cache. If this logical name is set to 0, the process cannot insert any rows into the row cache; the process can only use what is already there. If the logical name is set to 1, the process can insert new rows into the row cache, if they fit. This is a process logical name. The default is 1.
RDM$BIND_RCACHE_LATCH_SPIN_COUNT	LNM$FILE_DEV	INTEGER	Specifies how many times the RCS process should attempt to get the row cache latch before hibernating. This consumes CPU, but can acquire the latch faster. Recommended values are in the thousands. The default is 1024.
RDM$BIND_RCACHE_RCRL_COUNT	LNM$FILE_DEV	INTEGER	Specifies the number of rows to be reserved when empty slots are acquired in a row cache. This overrides the NUMBER OF RESERVE ROWS IS N clause. The default is 0.
RDM$BIND_RCS_BATCH_COUNT	LNM$SYSTEM_TABLE	INTEGER	Specifies the number of rows the RCS attempts to write during the course of a checkpoint or sweep operation. The default is 3000.

Table 4–3 (Cont.) Logical Names Specific to Row Caching

Logical Name	Table in Which Logical Is Stored	Value	Description
RDM$BIND_RCS_ CARRYOVER_ENABLED	LNM$SYSTEM_ TABLE	0 or 1	Specifies whether or not the RCS honors carryover locks for logical and physical areas. If the logical name is set to 0, the RCS does not honor carryover locks for logical and physical areas. It continues to hold them. (This enhances RCS performance, but prevents exclusive access to these logical and physical areas.) If the logical name is set to 1, the RCS honors carryover locks and releases logical and physical area locks that it is holding (due to a prior operation). The default is 1.
RDM$BIND_RCS_CKPT_ COLD_ONLY	LNM$SYSTEM_ TABLE	0 or 1	Specifies whether all marked records are included in a checkpoint or sweep operation. If the logical name is set to 0, checkpoint and sweep operations include all marked records in a row cache. If the logical name is set to 1, the only records included in a checkpoint operation are those that were marked before the prior checkpoint interval. (The only records included in the checkpoint operations are the older data. This will cause more of the .aij file to be read during a recovery operation). The default is 0.
RDM$BIND_RCS_CKPT_ BUFFER_CNT	LNM$SYSTEM_ TABLE	INTEGER	Specifies the number of buffers to use when writing records to backing files during checkpoint operations. The default is 15.
RDM$BIND_RCS_CLEAR_ GRICS_DBR_CNT	LNM$SYSTEM_ TABLE	INTEGER	Specifies the frequency (based on the number of DBR processes that run) at which the RCS attempts to release references in the row cache that were left by abnormally terminated processes. For each sweep request for a cache, if at least this number of DBR processes have run since the last sweep for the cache, the RCS will initiate a Release GRICs operation. This operation can have a minor performance impact to cache users and can also delay the RCS from performing other operations. This is why it is a periodic event. The maximum value of the logical name is 100,000. The default value is 25. Defining the logical name with a value of 0 disables the clearing of reference counts.

Table 4–3 (Cont.) Logical Names Specific to Row Caching

Logical Name	Table in Which Logical Is Stored	Value	Description
RDM$BIND_RCS_ CREATION_IMMEDIATE	LNM$SYSTEM_ TABLE	0 or 1	Specifies when the RCS process is created when the database is opened. If the database is set to open is automatic and the logical is set to 0, then the RCS process is created with the first reference to a row cache. If the database is set to open is automatic and the logical name is set to 1, the RCS process is created with the first attach. If the database is set to open is manual and the logical name is set to 1, then RCS is started immediately. The default is 0.
RDM$BIND_RCS_KEEP_ BACKING_FILES	LNM$SYSTEM_ TABLE	0 or 1	Specifies when the RCS creates and deletes backing files. If the logical name is set to 0, then the RCS creates backing files on each startup and deletes backing files on each shutdown. If the logical name is set to 1, the RCS retains backing files on shutdown and reuses them on startup. The default is 0.
RDM$BIND_RCS_LOG_ FILE	LNM$SYSTEM_ TABLE	File name	Specifies the location and name of the optional RCS process log file. If the logical name is not defined, then no RCS logging is done. Oracle Corporation recommends that logging be turned on. If a location is not specified along with the file name, then the log file is created in the same location as the database root file.
RDM$BIND_RCS_LOG_ HEADER	LNM$SYSTEM_ TABLE	0 or 1	Specifies whether header sections are included in the RCS log file. If the logical name is set to 0, header sections are not inserted in the RCS log file. If the logical name is set to 1, normal header sections are inserted into the RCS log file. The default is 1.
RDM$BIND_RCS_LOG_ REOPEN_SIZE	LNM$SYSTEM_ TABLE	INTEGER	This logical name represents the maximum block size of the RCS log file before the RCS opens a new log file. The default is 0.

Table 4–3 (Cont.) Logical Names Specific to Row Caching

Logical Name	Table in Which Logical Is Stored	Value	Description
RDM$BIND_RCS_PRIORITY	LNM$SYSTEM_TABLE	INTEGER	This logical name represents the base priority of the RCS process.
RDM$BIND_RCS_SWEEP_COUNT	LNM$SYSTEM_TABLE	INTEGER	This logical name represents the number of rows to sweep. It overrides the NUMBER OF SWEEP ROWS IS N clause.
RDM$BIND_RUJ_GLOBAL_SECTION_ENABLED	LNM$SYSTEM_TABLE	0 or 1	If the logical name is set to 0, do not place RUJ I/O buffers in global section so DBR can see them. If the logical name is set to 1, place RUJ I/O buffers in a global section so that the DBR can see them. The default is 1 if row caching is enabled. The default is 0 if row caching is disabled.

5

RMU New and Changed Features

This chapter describes the new and changed elements and statements for the Oracle RMU interface to Oracle Rdb release 7.1. It presents the information in reference format as it would appear in the *Oracle Rdb7 Oracle RMU Reference Manual*.

5.1 How to Read Syntax Diagrams

This manual uses the following conventions to present the syntax of Oracle RMU commands:

- Oracle RMU commands and qualifiers appear in initial capitals. Commands and qualifiers can be entered in uppercase, lowercase, or mixed-case type.
- Command parameters and variables are represented in lowercase type (with the exception of the RMU keyword, as noted in the first list item). These format elements represent parameters or variables for which you must supply a value.
- Horizontal ellipsis points (...) mean that you can enter additional information, such as parameters or qualifier arguments.
- Brackets ([ ]) enclose optional clauses from which you can choose one or none.
- Braces ({ }) enclose clauses from which you must choose one.
- A comma (,) separating qualifier arguments means that you can specify one or more arguments with the qualifier.

5.2 New and Changed RMU Statements

The following sections describe new and changed RMU statements; only the statements that have changed are included here.

If a statement is included here, the complete statement or clause introduction and format information is included. However, only those arguments, usage notes, and examples that are relevant to the new feature are documented here. See the *Oracle Rdb7 Oracle RMU Reference Manual* for the arguments, usage notes, and examples for features that existed prior to release 7.1.

RMU Backup Command

Creates a backup copy of the database and places it in a file. If necessary, you can later use the RMU Restore command to restore the database to the condition it was in at the time of the backup operation.

Format

RMU/Backup root-file-spec backup-file-spec

Command Qualifiers	Defaults
[No]Accept_Label	Noaccept_Label
[No]Acl	Acl
Active_IO=max-writes	Active_IO=3
Block_Size=integer	See description
[No]Checksum_Verification	Checksum_Verification
Crc[=Autodin_II]	See description
Crc=Checksum	See description
[No]Compression[=method]	Nocompression
Nocrc	See description
Density=number	See description
Disk_File[=(options)]	See description
Exclude[=storage-area[,...]]	See description
[No]Execute	See description
Extend_Quantity=number-block	Extend_Quantity=2048
[No]Group_Size=interval	See description
Include[=storage-area[,...]]	See description
[No]Incremental	Noincremental
Incremental=option	None
Journal=file-name	See description
Label=(label-name-list)	See description
List_Plan=output-file	See description
Loader_Synchronization	See description
Lock_Timeout=seconds	See description
[No]Log	See description
Master	See description
[No]Media_Loader	See description
No_Read_Only	See description
No_Worm	See description
[No]Online	Noonline
Owner=user-id	See description
Page_Buffers=number-buffers	Page_Buffers=2
Parallel=(Executor_Count=n[,options])	See description
Prompt={Automatic or Operator or Client}	See description
Protection[=file-protection]	See description
[No]Quiet_Point	Quiet_Point
Reader_Thread_Ratio=n	Reader_Thread_Ratio=5
Restore_Options=file-name	None
[No]Rewind	Norewind
[No]Scan_Optimization	See description
Tape_Expiration=date-time	Current time

Command Parameters

backup-file-spec

The file specification for the backup file. The default file extension is .rbf. Depending on whether you are performing a backup operation to magnetic tape, disk, or multiple disks, the backup file specification should be specified as follows:

- If you are backing up to magnetic tape
  - Oracle Corporation recommends that you supply a backup file name that is 17 or fewer characters in length. File names longer than 17 characters might be truncated.
  - If you use multiple tape drives, the backup-file-spec parameter must be provided with (and only with) the first tape drive name. Additional tape drive names must be separated from the first and subsequent tape drive names with commas.
- If you are backing up to multiple or single disk files
  - It is good practice to write backup files to a device other than the devices where the database root, storage area, and snapshot files of the database are located. This way, if there is a problem with the database disks, you can still restore the database from a backup file.
  - If you use multiple disk files, the backup-file-spec parameter must be provided with (and only with) the first disk device name. Additional disk device names must be separated from the first and subsequent disk device names with commas. You must include the Disk_File qualifier. For example:

    ```
    $ RMU/BACKUP/DISK_FILE MF_PERSONNEL.RDB DEVICE1:[DIRECTORY1]MFP.RBF,
     _$ DEVICE2:[DIRECTORY2]
    ```

    As an alternative to listing the disk device names on the command line (which, if you use several devices, can exceed the line-limit length for a command line), you can specify an options file in place of the backup-file-spec. For example:

    ```
    $ RMU/BACKUP/DISK_FILE LARGE_DB "@DEVICES.OPT"
    ```

    The contents of devices.opt might appear as follows:

    ```
    DEVICE1:[DIRECTORY1]LARGE_DB.RBF
    DEVICE2:[DIRECTORY2]
    ```

    The resulting backup files created from such an options file would be:

```
DISK1:[DIRECTORY1]LARGE_DB.RBF
DISK2:[DIRECTORY2]LARGE_DB01.RBF
```

Note that the same directory must exist on each device before you issue the command. Also, if you forget to specify the Disk_File qualifier, you receive an error message similar to the following:

```
$RMU/BACKUP MF_PERSONNEL DEVICE1:[DIRECTORY1]MFP.RBF, -
DEVICE2:[DIRECTORY2]
%RMU-F-NOTBACFIL, DEVICE1:[DIRECTORY1]MFP.RBF; is not a valid backup
file
%RMU-F-FTL_BCK, Fatal error for BACKUP operation at  2-MAY-2001
09:44:57.04
```

Command Qualifiers

Compression
Compression=LZSS
Compression=Huffman
Nocompression

Allows you to specify the compression method to use before writing data to the backup file. This reduces performance, but may be justified when the backup file is a disk file, or is being backed up over a busy network, or is being backed up to a tape drive that does not do its own compression. You probably do not want to specify the Compression qualifier when you are backing up a database to a tape drive that does its own compression; in some cases doing so can actually result in a larger file.

If you specify the Compression qualifier without a value, the default is Compression=LZSS.

Disk_File[=(Max_File_Size=number-megabytes, Writer_Threads=integer)]

Specifies that you want to perform a multithreaded backup operation to disk files, floppy disks, or other disks external to the PC.

The Max_File_Size keyword forces RMU to start a new backup file when the current file reaches the size, in megabytes, that you specify with this keyword. This is useful when your plan is to create the backup files on disk and then copy these files to another medium (such as a writable CD-ROM). By default, there is no maximum size for a backup file to disk.

The Writer_Threads keyword specifies the number of threads that Oracle RMU should use when performing a multithreaded backup operation to disk files. You can specify no more than one writer thread per device specified on the command line (or in the command parameter options file). By default, one writer thread is used.

This qualifier and all qualifiers that control tape operations (Accept_Label, Density, Label, Loader_Synchronization, Master, Media_Loader, Rewind, and Tape_Expiration) are mutually exclusive.

Exclude[=storage-area[,...]]
Specifies the storage areas that you want to exclude from the backup file. If you specify neither the Exclude nor the Include qualifier with the RMU Backup command, or if you specify the Exclude qualifier but do not specify a list of storage area names, a full and complete backup operation is performed on the database. This is the default behavior.

Specifying a list of storage area names with the Exclude qualifier tells Oracle RMU to exclude those storage areas from the backup file but to include all of the other storage areas. If you specify more than one database storage area in the Exclude qualifier, place a comma between each storage area name and enclose the list of names within parentheses.

Use the Exclude=* qualifier to indicate that you want only the database root file to be backed up. Note that a backup file created with the Exclude=* qualifier can be restored only with the RMU Restore Only_Root command.

You can use an indirect command file as shown in the following example:

```
$ RMU/BACKUP/EXCLUDE="@EXCLUDE_AREAS.OPT" -
_$ MF_PERSONNEL.RDB PARTIAL_MF_PERS.RBF
%RMU-I-NOTALLARE, Not all areas will be included in this backup file
```

If you use the Exclude qualifier with a list of storage area names, your backup file will be a by-area backup file because the Exclude qualifier causes database storage areas to be excluded from the backup file. The following example shows the informational message you receive if you do not back up all of the areas in the database:

```
%RMU-I-NOTALLARE, Not all areas will be included in this backup file
```

By using the RMU Backup and RMU Restore commands, you can back up and restore selected storage areas of your database. This Oracle RMU backup and restore by-area feature is designed to:

- Speed recovery when corruption occurs in some (not all) of the storage areas of your database

- Reduce the time needed to perform backup operations because some data (data in read-only storage areas, for example) does not need to be backed up with every backup operation performed on the database

If you plan to use the RMU Backup and RMU Restore commands to back up and restore only selected storage areas for a database, you should perform full and complete backup operations on the database at regular intervals.

If you plan to back up and restore only selected storage areas of a database, Oracle Corporation also strongly recommends that you enable after-image journaling for the database. This ensures that you can recover all of the storage areas in your database if a system failure occurs.

If you *do not* have after-image journaling enabled and one or more of the areas restored with the RMU Restore command are not consistent with the unrestored storage areas, Oracle Rdb does not allow any transaction to use the storage areas that are not consistent in the restored database. In this situation, you can return to a working database by restoring the database, using the backup file from the last full and complete backup operation of the database storage areas. However, any changes made to the database since the last full and complete backup operation are not recoverable.

If you *do* have after-image journaling enabled, use the RMU Recover command (or the Restore command with the Recover qualifier) to apply transactions from the .aij file to storage areas that are not consistent after the RMU Restore command completes; that is, storage areas that are not in the same state as the rest of the restored database. You cannot use these areas until you recover the database. When the RMU Recover command completes, your database will be consistent and usable.

Using the Exclude or Include qualifier gives you greater flexibility for your backup operations, along with increased file management and recovery complexity. Users of large databases might find the greater flexibility of the backup operation to be worth the cost of increased file management and recovery complexity.

You receive the CONFLSWIT error message if you specify Exclude=area-list and Include=area-list in the same RMU Backup command.

Include[=storage-area[,...]]

Specifies storage areas that you want to include in the backup file. If you specify neither the Include nor the Exclude qualifier with the RMU Backup command, a full and complete backup operation is performed on the database by default. You can specify the Include=* qualifier to indicate that you want all storage areas included in the backup file, but this is unnecessary because this is the default behavior. The default behavior is performed also when you specify the Include qualifier without specifying a list of storage area names.

Specifying a list of storage area names with the Include qualifier tells Oracle RMU to include those storage areas in the backup operation but to exclude all of the other storage areas. If you specify more than one database storage area in the Include qualifier, place a comma between each storage area name and enclose the list of names within parentheses.

You receive the CONFLSWIT error message if you specify the Exclude=area-list and the Include=area-list qualifiers in one RMU Backup command.

If you use the Include qualifier, your backup operation will be a by-area backup operation because the areas not specified with the Include qualifier are excluded from the backup file. If you do not back up all of the areas in the database, you receive the following informational message:

```
%RMU-I-NOTALLARE, Not all areas will be included in this backup file
```

By using the RMU Backup and RMU Restore commands, you can back up and restore selected storage areas of your database. This Oracle RMU backup and restore by area feature is designed to:

- Speed recovery when corruption occurs in some (not all) of the storage areas of your database
- Reduce the time needed to perform backup operations because some data (data in read-only storage areas, for example) does not need to be backed up with every backup operation performed on the database

See the description of the Exclude qualifier for information on the implications of using these commands to back up and restore selected areas of your database.

The Include qualifier can be used with indirect file references.

Prompt=Automatic
Prompt=Operator
Prompt=Client

Specifies where server prompts are to be sent. When you specify Prompt=Automatic, then prompts are sent to the standard input device, and when you specify Prompt=Operator, prompts are sent to the server console. When you specify Prompt=Client, prompts are sent to the client system.

Reader_Thread_Ratio=integer

Specifies the number of reader threads to be created per output thread. (Prior to Oracle Rdb release 7.1, one backup reader thread was created for each storage area in the database. For large databases, that sometimes created an unwieldy number of threads.) At most, one reader thread is assigned to a storage area and one writer thread is assigned to each backup device. If your database has numerous small storage areas, increasing the reader thread ratio may improve performance.

Usage Notes

- When you use more than one tape drive for a backup operation, ensure that all of the tape drives are the same type (for example, all of the tape drives must be TA90s or TZ87s or TK50s). Using different tape drive types (for example, one TK50 and one TA90) for a single database backup operation may make database restoration difficult or impossible.

  Oracle RMU attempts to prevent you from using different tape drive densities during a backup operation but is not able to detect all invalid cases and expects that all tape drives for a backup are of the same type.

  As long as all of the tapes used during a backup operation can be read by the same type of tape drive during a restore operation, the backup is likely to be valid. This may be the case, for example, when you use a TA90 and a TA90E.

  Oracle Corporation recommends that, on a regular basis, you test your backup and recovery procedures and environment using a test system. You should restore the database and then recover using after-image journals (AIJs) to simulate failure recovery of the production system.

  Consult the *Oracle Rdb7 Guide to Database Maintenance*, the *Oracle Rdb7 Guide to Database Design and Definition*, and the *Oracle Rdb7 Oracle RMU Reference Manual* for additional information about Oracle Rdb backup and restore operations.

- Oracle Rdb cannot continue a single .rda file across multiple disks. This means that, during a multidisk backup operation, each device must have enough free space to hold the largest storage area in the database. If the storage areas are on stripe sets and are larger than any actual single disk, then the devices specified for the backup file must be striped also.

  It is not possible to indicate which storage area should be backed up to a given device.

- The default behavior of the Checksum_Verification qualifier for the RMU Backup command is changed with this release of Oracle Rdb.

  In previous releases, the default value for the Checksum_Verification qualifier was Nochecksum_Verification. Beginning with this release, the default value is Checksum_Verification. The Checksum_Verification qualifier requests that the RMU Backup command verify the checksum stored on each database page before the backup operation is applied, thereby providing end-to-end error detection on the database I/O.

Oracle Corporation recommends that you accept the new default behavior for your applications. The new default behavior prevents you from including corrupt database pages in backup files and optimized .aij files. Without the checksum verifications, corrupt data pages in these files are not detected when the files are restored. The corruptions on the restored page may not be detected until weeks or months after the backup file is created, or it is possible the corruption may not be detected at all.

If you prefer the Nochecksum_Verification behavior, be aware of the risk of undetected corruptions.

Examples

Example 1 Performing a Multidisk Backup Operation

```
$ RMU/BACKUP/DISK MF_PERSONNEL DEVICE1:[DIRECTORY1]MFP.RBF,
_$ DEVICE2:[DIRECTORY2]
.
.
.
%RMU-I-COMPLETED, BACKUP operation completed at  1-MAY-2001 17:40:53.81
```

RMU Backup After_Journal Command

Creates a backup file of the database after-image journal (.aij) file or files.

Oracle Rdb supports two types of after-image journaling mechanisms: one that employs a single, extensible .aij file and another that employs multiple, fixed-size .aij files. The type of journaling mechanism being used at the time the backup operation starts can affect how you should specify the backup command. Further information on how these two journaling mechanisms affect the backup operation appears in the *Oracle Rdb7 Oracle RMU Reference Manual*.

The backup .aij file is an actual, usable .aij file that can be applied to the appropriate Oracle Rdb database in a recovery operation. The RMU Backup After_Journal command can be used while users are attached to the database.

Format

RMU/Backup/After_Journal root-file-spec {backup-file-spec or " "}

Command Qualifiers	Defaults
[No]Accept_Label	Accept_Label
Active_IO=max-writes	Active_IO=3
Block_Size=integer	See description
[No]Continuous=(n)	Nocontinuous
[No]Crc	See description
Crc[=Autodin_II]	See description
Crc=Checksum	See description
Density=number	See description
[No]Edit_Filename=(options)	Noedit_Filename
Format={Old_File or New_Tape}	Format=Old_File
[No]Group_Size[=interval]	See description
[No]Interval=number-seconds	Nointerval
Label=(label-name-list)	See description
Lock_Timeout=seconds	See description
[No]Log	See description
[No]Media_Loader	See description
Owner=user-id	See description
Prompt={Automatic or Operator or Client}	See description
Protection=file-protection	See description
[No]Quiet_Point	Quiet_Point
[No]Rename	Norename
[No]Rewind	Norewind
[No]Sequence=(n,m)	Nosequence
Tape_Expiration=date-time	Current time
[No]Threshold=disk-blocks	Nothreshold
Until=date-time	None
[No]Wait=n	See description

Command Parameters

backup file-spec
The file specification for the .aij backup file. The default file extension is .aij unless you specify the Format=New qualifier. In this case, the default file extension is .aij_rbf.

Command Qualifiers

Prompt=Automatic
Prompt=Operator
Prompt=Client
Specifies where server prompts are to be sent. When you specify Prompt=Automatic, then prompts are sent to the standard input device, and when

you specify Prompt=Operator, prompts are sent to the server console. When you specify Prompt=Client, prompts are sent to the client system.

Usage Notes

- Previously, the Rename qualifier was only valid for fixed-size .aij files. Beginning with this release of Oracle Rdb, you can use the Rename qualifier with extensible .aij files.

RMU Copy_Database Command

Permits you to copy a database.

Format

RMU/Copy_Database root-file-spec [storage-area-list]

Command Qualifiers	Defaults
[No]After_Journal[=file-spec]	See description
[No]Aij_Options=journal-opts-file	See description
[No]Cdd_Integrate	Nocdd_Integrate
[No]Checksum_Verification	Checksum_Verification
Directory=directory-spec	None
[No]Duplicate	Noduplicate
Global_Buffers=global-buffer-options	Current value
Local_Buffers=local-buffer-options	Current value
Lock_Timeout=n	See description
[No]Log	See description
Nodes_Max=n	Current value
[No]Online	Noonline
Open_Mode=(Automatic or Manual)	Current value
Options=file-spec	None
Page_Buffers=n	Page_Buffers=3
Path=cdd-path	Current value
Quiet_Point	Quiet_Point
Root=file-spec	None
Transaction_Mode=(modes-list)	Transaction_Mode=Current
Users_Max=n	Current value

File or Area Qualifiers	Defaults
Blocks_Per_Page=n	None
Extension={Disable or Enable}	Current value
File=file-spec	None
Read_Only	Current value
Read_Write	Current value
Snapshots=(Allocation=n, File=file-spec)	None
[No]Spams	Current value
Thresholds=(n,n,n)	None
[No]Worm	Current value

Command Qualifiers

Transaction_Mode=(modes-list)

Sets the allowable transaction modes for the database root file created by the copy operation. The modes-list can include one or more of the following transaction modes:

- All - Enables all transaction modes.
- Current - Enables all transaction modes that are set for the source database. This is the default transaction mode.
- None - Disables all transaction modes.
- [No]Batch_Update
- [No]Read_Only
- [No]Exclusive
- [No]Exclusive_Read
- [No]Exclusive_Write
- [No]Protected
- [No]Protected_Read
- [No]Protected_Write
- [No]Read_Write
- [No]Shared
- [No]Shared_Read
- [No]Shared_Write

Your copy operation must include the database root file. Otherwise, RMU returns the CONFLSWIT error when you issue an RMU Copy_Database command with the Transaction_Mode qualifier.

If you specify more than one transaction mode in the modes-list, enclose the list in parentheses and separate the transaction modes from one another with a comma. Note the following:

- When you specify a negated transaction mode, it indicates that a mode is not an allowable access mode. For example, if you specify the Noexclusive_Write access mode, it indicates that exclusive write is not an allowable access mode for the copied database.

- If you specify the Shared, Exclusive, or Protected transaction mode, Oracle RMU assumes that you are referring to both reading and writing in that transaction mode.
- No mode is enabled unless you add that mode to the list, or you use the All option to enable all transaction modes.
- You can list one transaction mode that enables or disables a particular mode followed by another that does the opposite.

  For example, Transaction_Mode=(Noshared_Write, Shared) is ambiguous because the first value disables Shared_Write access and the second value enables Shared_Write access. Oracle RMU resolves the ambiguity by first enabling the modes as specified in the modes-list and then disabling the modes as specified in the modes-list. The order of items in the list is irrelevant. In the example presented previously, Shared_Read is enabled and Shared_Write is disabled.

Usage Notes

No new usage notes.

Examples

Example 1 Setting the Allowable Transactions Modes for a Copied Database

The following example demonstrates how to disallow exclusive mode for a copied database. It then shows the error messages returned when a user attempts to access the copied database using the disallowed mode:

```
$ rmu/copy_database/transaction_mode=noexclusive/directory=[.copy] -
_$ mf_personnel.rdb
%RMU-W-DOFULLBCK, full database backup should be done to ensure future
recovery
$ SQL
SQL> ATTACH 'FILENAME mf_personnel.rdb';
SQL> SET TRANSACTION READ WRITE RESERVING EMPLOYEES FOR EXCLUSIVE WRITE;
%RDB-E-BAD_TPB_CONTENT, invalid transaction parameters in the transaction parame
ter block (TPB)
-RDMS-E-INVTRANOPT, the transaction option "EXCLUSIVE WRITE" is not allowed
SQL>
```

RMU Dump After_Journal Command

Displays an after-image journal (.aij) file, a backed up .aij file (.aij if the backup is on disk, .aij_rbf if the .aij file was backed up to tape), or an optimized after-image journal (.oaij) file in ASCII format. Use this command to examine the contents of your .aij, .aij_rbf, or .oaij file. Whenever the term .aij file is used in this RMU Dump After_Journal command description, it refers to .oaij and .aij_rbf files, as well as .aij files.

An .aij file contains header information and data blocks. Header information describes the data blocks, which contain copies of data stored in the database file.

Format

RMU/Dump/After_Journal aij-file-name

Command Qualifiers	Defaults
Active_IO=max-reads	Active_IO=3
[No]Data	Data
End=integer	See description
First=(select-list)	See description
Format={Old_File or New_Tape}	Format=Old_File
Label=(label-name-list)	See description
Last=(select=list)	See description
[No]Media_Loader	See description
Options={Statistics or Nostatistics}	Option=Statistics
Output=file-name	See description
Prompt={Automatic or Operator or Client}	See description
[No]Rewind	Norewind
Start=integer	See description
State=Prepared	See description

Command Qualifiers

First=select-list
Last=select-list
Allows you to specify where you want the dump output to begin and end. If you specify more than one keyword in the select-list, separate the keywords with commas and enclose the list in parentheses. If you specify multiple items in the select list, the first occurrence is the one that will activate Oracle RMU. For example,

if you specify `First=(Block=100,TSN=0:52)`, the dump will start when either block 100 or TSN 52 is encountered.

The First and Last qualifiers are optional. You can specify both, either, or neither of them. The keywords specified for the First qualifier can differ from the keywords specified for the Last qualifier.

The select-list of these qualifiers consists of a list of one or more of the following keywords:

- RECORD=record#

  Specifies the first or last record in the AIJ journal. This is the same as the existing Start and End qualifiers, which are still supported but obsolete.

- BLOCK=block#

  Specifies the first or last block in the AIJ journal.

- TSN=tsn

  Specifies the first or last TSN in the AIJ journal, using the standard [n:]m TSN format.

- TID=tid

  Specifies the first or last TID in the AIJ journal.

- TIME=date_time

  Specifies the first or last date and time in the AIJ journal, using the standard date-time format.

By default, the entire .aij file is dumped.

Prompt=Automatic
Prompt=Operator
Prompt=Client
Specifies where server prompts are to be sent. When you specify Prompt=Automatic, then prompts are sent to the standard input device, and when you specify Prompt=Operator, prompts are sent to the server console. When you specify Prompt=Client, prompts are sent to the client system.

Usage Notes

The First and Last qualifiers have been added to make dumping portions of the .aij file easier. The Start and End qualifiers were intended to provide similar functionality, but are difficult to use because you seldom know, nor can you determine, the AIJ record number prior to issuing the command.

Examples

Example 1 Starting a Dump from Block 100 or TSN 52, Whichever Occurs First

```
$ RMU/DUMP/AFTER_JOURNAL /FIRST=(BLOCK=100,TSN=0:52) MF_PERSONNEL.AIJ
```

RMU Dump Backup_File Command

Displays or writes to a specified output file the contents of a backup file. Use this command to examine the contents of a backup (.rbf) file created by the RMU Backup command.

Format

RMU/Dump/Backup_File backup-file-spec

Command Qualifiers	Defaults
Active_IO=max-reads	Active_IO=3
Disk_File[=(Reader_Threads=n)]	See description
Journal=file-name	See description
Label=(label-name-list)	See description
[No]Media_Loader	See description
Options=options-list	See description
Output=file-name	See description
Process=process-list	See description
Prompt={Automatic or Operator or Client}	See description
Restore_Options=file-name	None
[No]Rewind	Norewind
Skip=skip-list	See description

Command Qualifiers

Disk_File=[(Reader_Threads=integer)]

Specifies that you want to dump a multiple disk backup file. This is a backup file that was created by the RMU Backup command with the Disk_File qualifier.

The Reader_Threads keyword specifies the number of threads that Oracle RMU should use when performing a multithreaded read operation from disk files. You can specify no more than one reader thread per device specified on the command line (or in the command parameter options file). By default, one reader thread is used.

This qualifier and all qualifiers that control tape operations (Label, Media_Loader, and Rewind) are mutually exclusive.

Prompt=Automatic
Prompt=Operator
Prompt=Client
Specifies where server prompts are to be sent. When you specify Prompt=Automatic, prompts are sent to the standard input device, and when you specify Prompt=Operator, prompts are sent to the server console. When you specify Prompt=Client, prompts are sent to the client system.

Usage Notes

No new usage notes.

RMU Extract Command

Reads and decodes Oracle Rdb metadata and reconstructs equivalent statements in RDO (Relational Database Operator) or SQL (structured query language) code for the definition of that database. These statements can either be displayed or extracted. You can use these statements to create your database again if you no longer have the RDO or SQL code that defined your database. (The RDO interface is available only on OpenVMS platforms.)

In addition, you can direct the RMU Extract command to produce output for the following:

- The Oracle Expert for Rdb Physical Design Language (PDL) Volume Script (Items=Volume)
- An SQL or RDO IMPORT script (Items=Import)
- An RMU Unload command for each table (Items=Unload)
- An RMU Load command for each table (Items=Load)
- An RMU Set Audit command for a database on OpenVMS (Items=Security)
- An RMU Verify command for the database (Items=Verify)

Format

RMU/Extract root-file-spec

Command Qualifiers	Defaults
Items[=item-list]	Items=All
Language=lang-name	Language=SQL
[No]Log[=log-file]	Nolog
Options=options-list	Option=Normal
[No]Output[=out-file]	See description

Command Qualifiers

Items[=item-list]

Allows you to extract and display selected definitions.

The following items are introduced in Oracle Rdb release 7.1:

- Role

Displays role definitions as defined by the SQL CREATE ROLE or ALTER ROLE command. In addition, any roles that have been granted are displayed as a GRANT statement. By default, roles are not extracted, nor are they included when you specify the Items=All qualifier.

- Sequence

  Displays sequence definitions as defined by the SQL CREATE SEQUENCE or ALTER SEQUENCE command. Sequences are included by default and are included when you specify the Items=All qualifier. However, system sequences (used for the SQL CREATE USER and CREATE ROLE statements) are not extracted.

- User

  Displays user definitions as defined by the SQL CREATE USER or ALTER USER command. In addition, if you also specify Role with the Item qualifier, then any roles that have been granted are displayed as a GRANT statement. By default, Users are not displayed, nor are they displayed when you specify the Items=All qualifier.

Options=options-list

This qualifier is used to change the output of the RMU Extract command. This qualifier is not applied to output created by the Items=Volume, Items=Unload, Items=Load, Items=Security, or the Items=Verify qualifier.

The following options are introduced in Oracle Rdb release 7.1:

- Order_By_Name and Noorder_By_Name

  Order_By_Name displays the storage area, cache, and journal names for the items Database, Alter_Database (also known as Change_Database), and Import in alphabetic order by the ASCII collating sequence.

  Noorder_By_Name displays the storage area, cache, and journal names for the items Database, Alter_Database, and Import in approximate definition order. The default ordering is approximate because a DROP STORAGE AREA, DROP CACHE, or DROP JOURNAL statement frees a slot that can be reused, thus changing the order. The Noorder_By_Name option is the default.

  Note: If the identifier character set for the database is not MCS or ASCII, then this option is ignored. Characters from other character sets do not sort appropriately under the ASCII collating sequence.

- Omit_Disabled and Noomit_Disabled

  The Omit_Disabled option causes all disabled objects to be omitted from the output of the RMU Extract command. This includes indexes that have MAINTENANCE IS DISABLED, USERS with ACCOUNT LOCK, and disabled triggers and constraints.

  The Noomit_Disabled option causes all disabled objects to be included in the output from the RMU Extract command. The Noomit_Disabled option is the default.

- Disable_Objects and Nodisable_Objects

  The Disable_Objects option requests that all disabled objects be written to the RMU Extract output file as disabled (see the description for the Omit_Disabled option). The Nodisable_Objects option displays the objects but omits the disabling syntax. The Disable_Objects option is the default.

- Match:match-string

  The MATCH option allows selection of wildcard object names from the database. The match-string can contain the standard SQL wildcard characters: the percent sign (%) to match any number of characters, and the underscore (_) to match a single character. In addition, the backslash (\) can be used to prefix these wildcards to prevent them from being used in matching. If you are matching a literal backslash, use the backslash twice, as shown in the following example:

  ```
  Option=Match:"A1\\A2%"
  ```

  The match-string defaults to the percent sign (%) so that all objects are selected. To select those objects that start with JOB, use the qualifier Option=Match:"JOB%".

  From the mf_personnel database, this command displays the definitions for the domains JOB_CODE_DOM and JOB_TITLE_DOM, the tables JOBS and JOB_HISTORY, the index JOB_HISTORY_HASH, and the storage maps JOBS_MAP and JOB_HISTORY_MAP.

  The match string can be quoted as shown if the string contains spaces or other punctuation characters used by DCL or other command language interfaces. Most object names are space filled; therefore, follow the match-string with the percent sign (%) to match all trailing spaces.

  The Match option can be used in conjunction with the Item qualifier to extract specific tables, indexes, and so on, based on their name and type.

Usage Notes

No new usage notes.

RMU Load Command

You can use the RMU Load command to:

- Perform the initial load of an Oracle Rdb database.
- Reload a table after performing a restructuring operation.
- Load an archival database.
- Move data from one database to another.
- On OpenVMS, load security audit records from an OpenVMS security audit table into the database being audited, or into a different database than the one being audited.
- Load additional rows into an existing table. (However, note that you cannot use the RMU Load command to modify existing rows.)
- Import data into a database from an application that generates RMS files.

You can load data using either of the following two methods:

- A single-process method

  This was the only method available prior to Oracle Rdb release 7.0. The single-process method uses one process to both read the input file and load the target table.

- A multiprocess method, also called a parallel load

  This method was new in Oracle Rdb release 7.0. The **parallel load** method, which you specify with the Parallel qualifier, enables Oracle RMU to use your process to read the input file and use one or more executors (subprocesses or detached slave process, depending on additional factors) to load the data into the target table. This results in concurrent read and write operations, and in many cases, substantially improves the performance of the load operation.

By default, Oracle RMU sets up a parallel load operation as follows:

- Your process serves as the load operation execution manager.
- Each storage area (partition) in the table being loaded is assigned an executor.
- Each executor is assigned four communications buffers.

  (You can override this default with the Buffer_Count option to the Parallel qualifier.)

- Each communications buffer holds the number of rows defined by the Row_Count qualifier.

Once the executors and communications buffers are set up, the parallel load operation processes the input file as follows:

1. Your process begins reading the input file and determines the target storage area for each row in the input file.
2. Your process places each row in the communications buffer for the executor assigned to the data's target storage area.
3. When an executor's first communications buffer becomes full, it begins loading the data into the target storage area.
4. If your process has another portion of data ready for a given executor before that executor has completed loading its first buffer of data, your process places the next portion of data in the second communications buffer for that executor.
5. Each executor, concurrent with each of the other executors, loads the data from its buffers.
6. Your process continues reading, sorting, and assigning data to each executor (by placing it in that executor's communications buffer) until all of the data from the input file has been sorted, assigned, and loaded.

The Row_Count qualifier and Parallel qualifier (which provides the Executor_Count and Buffer_Count options) give you the ability to fine-tune the Parallel load operation.

See the *Oracle Rdb7 Guide to Database Design and Definition* for tips on optimizing the performance of the load operation.

Format

RMU/Load root-file-spec table-name input-file-name

Command Qualifiers	Defaults
Audit[=Database_File=db-name]	No audit table loaded
Buffers=n	See description
Commit_Every=n	See description
[No]Constraints[=Deferred]	Constraints
Corresponding	See description
[No]Defer_Index_Updates	Nodefer_Index_Updates
[No]Execute	Execute
Fields=(column-name-list)	See description
List_Plan=output-file	See description
[No]Log_Commits	Nolog_Commits
Parallel[=(options)]	See description
[No]Place	Noplace
Record_Definition=({File or Path}=name[,options])	See description
[No]Restricted_Access	Norestricted_Access
Row_Count=n	See description
[No]Skip	Noskip
Statistics=(stat-opts)	See description
Transaction_Type=share-mode	Transaction_Type=Protected
[No]Trigger_Relations[=(table-name-list)]	Trigger_Relations

Command Qualifiers

Restricted_Access
Norestricted_Access

Allows a single process to load data and enables some optimizations available only when restricted access is in use.

If you are loading a table from an RMU Unload file that contains LIST OF BYTE VARYING data, then the Restricted_Access qualifier reserves the LIST areas for EXCLUSIVE access. This reduces the virtual memory used by long transactions during a load operation and also eliminates I/O to the snapshot files for the LIST storage areas.

The Restricted_Access and Parallel qualifiers are mutually exclusive and cannot be specified together on the same RMU Load command line or within a plan file. While RMU Load is running with the Restricted_Access qualifier specified, no other user can attach to the database.

The default is the Norestricted_Access qualifier.

Usage Notes

No new usage notes.

RMU Open Command

Opens a database root file and maps its global section to the contents of an OpenVMS virtual address file.

You can use the RMU Open command in conjunction with the SQL ALTER DATABASE statement to control access to the database. See the description of the OPEN IS {AUTOMATIC | MANUAL} clause of the SQL ALTER DATABASE statement in the *Oracle Rdb7 SQL Reference Manual* for details.

Format

RMU/Open root-file-spec [,...]

Command Qualifiers	Defaults
Access=[Un]Restricted	See description
Global_Buffers[=(Total=i,User_Limit=j)]	See description
Path	None
Row_Cache=Disabled	Row caching is enabled
[No]Wait	Nowait

Command Qualifiers

Row_Cache=Disabled
Disables row caching. This qualifier is provided for use with hot standby databases. Row caching cannot be enabled on a hot standby database while replication is active. If it is enabled, the hot standby feature will not start.

This restriction exists because rows in the row cache are accessed through logical dbkeys. However, information transferred to the hot standby database through the after-image journal facility only contains physical dbkeys. Because there is no way to maintain rows in the cache through the hot standby processing, you must disable row caching when the standby database is open and replication is active.

Wait
Nowait
Specifies whether the system prompt should be returned before the database is completely open and available. Specify the Wait qualifier if you want the system prompt returned when the database is completely open and available. Specify

Nowait if you want the system prompt returned immediately, regardless of the state of the open operation.

The Nowait qualifier is the default.

Usage Notes

No new usage notes.

Examples

Example 1 Disabling Row Caching

```
$ RMU/OPEN MF_PERSONNEL.RDB /ROW_CACHE=DISABLED
```

RMU Recover Command

Completes a database reconstruction by processing past transactions from the after-image journal (.aij) file or optimized after-image journal (.oaij) file against a database restored from a backup file.

Format

RMU/Recover aij-file-name-list

Command Qualifiers	Defaults
Active_IO=max-rate	Active_IO=3
Aij_Buffers=integer	Aij_Buffers=20
Areas[=storage-area[,...]]	All storage areas
[No]Automatic	Automatic
Format={Old_File or New_Tape}	Format=Old_File
Just_Corrupt	See description
Label=(label-name-list)	See description
[No]Log	See description
[No]Media_Loader	See description
[No]Online	Noonline
Output=file-name	See description
Prompt={Automatic or Operator or Client}	See description
Resolve	See description[1]
No]Rewind	Norewind
Root=root-file-name	See description
[No]Trace	See description
Until="date-time"	Current time

[1] See the section on the RMU Recover Resolve command in the *Oracle Rdb7 Oracle RMU Reference Manual*.

Command Parameters

aij-file-name-list

A list of after-image journal (.aij) files to be applied to the database. You can supply this list using one of three methods, as follows:

- List the .aij files on the command line in the order in which they were created. In other words, the oldest .aij file must be the first in the list.

- Append all your .aij files into one file and supply a single .aij file name. However, you must be certain when you append the files that you append them in the order in which they were created.
- Use an indirect command file. Include an .aij file name on each line of the command file. If the number of .aij files needed for recovery is large, listing each one on the command line can exceed the maximum allowed command-line length. By using an indirect command file, you avoid this problem.

Command Qualifiers

Prompt=Automatic
Prompt=Operator
Prompt=Client
Specifies where server prompts are to be sent. When you specify Prompt=Automatic, then prompts are sent to the standard input device, and when you specify Prompt=Operator, prompts are sent to the server console. When you specify Prompt=Client, prompts are sent to the client system.

Usage Notes

No new usage notes.

RMU Repair Command

Corrects several types of database problems. You can use the RMU Repair command to:

- Repair all types of space area management (SPAM) page corruptions by reconstructing the SPAM pages in one or more storage areas.
- Repair all area bit map (ABM) page format errors.
- Repair all page tail errors to the satisfaction of the RMU Verify operation by making sure that every database page is in a logical area and contains the appropriate information for that logical area.
- Correct some performance problems that might otherwise have to be corrected by exporting and importing the database.
- Set damaged or missing segmented string (LIST OF BYTE VARYING) areas that are stored in write-once areas to null.

The RMU Repair command cannot correct corrupted user data, or corrupted indexes; use other commands such as RMU Restore, RMU Recover, RMU Load, or SQL IMPORT and delete the affected structures to correct these problems.

Note: Use of the Abm or the Initialize=Tsns qualifier disables after-image journaling. After issuing an RMU Repair command with these qualifiers, back up the database and reenable journaling manually.

Format

RMU/Repair root-file-spec

Command Qualifiers	Defaults
[No]Abm	Noabm
[No]All_Segments	All segments
Areas[={storage-area-list or *}]	See description
Checksum	See description
[No]Initialize=initialize-options	Noinitialize
[No]Spams	Spams
Tables[=table-list]	All nonsystem tables
Worm_Segments	None

Command Qualifiers

Initialize=initialize-options
Noinitialize

Allows you to specify initialization options. If more than one option is specified, separate the options with a comma, and enclose the list of options within parentheses. Note that after-image journaling is disabled when you specify the Initialize qualifier.

A new item for the Larea_Parameters=options-file initialize option is introduced in Oracle Rdb release 7.1.

The Larea_Parameters=options-file option specifies an options file (default file extension .opt) that contains a list of logical areas and parameter values that Oracle RMU uses to update the area inventory page (AIP) before it builds the space area management (SPAM) pages.

The Larea_Parameters options file contains lines in the following format on Oracle Rdb for OpenVMS:

```
name [/Areas=name][/Delete][/[No]Thresholds=(n[,n[,n]])[/Length=n]
```

A comment can be appended to the line (an exclamation point (!) is the comment character), and a line can be continued (as in DCL) by ending it with a hyphen (-).

The logical area can be specified by name or identification number (ID). The logical area named must be present in the AIP, or an error is generated.

The new Larea_Parameters options are the Type options. By specifying a Type option, you can update the on-disk logical area type in the AIP. For databases created prior to Oracle Rdb release 7.0.1, the logical area type information in the AIP is unknown. However, the Show Statistics utility depends on this information to display information on a per-logical-area basis. A logical area is a table, B-tree index, hash index, or any partition of one of these. Although the Show Statistics utility will prompt you to enter a logical area type when needed, the utility does not update the database AIP pages with the data that you supply.

Therefore, to update the on-disk logical area type in the AIP, you need to specify the type as follows:

- Type=Table

  Specifies that the logical area is a data table, such as is created with the SQL CREATE TABLE statement.

- Type=Btree

Specifies that the logical area is a B-tree index, such as is created with the SQL CREATE INDEXES TYPE IS SORTED statement.

- Type=Hash

  Specifies that the logical area is a hash index, such as is created with the SQL CREATE INDEX TYPE IS HASHED statement.

- Type=System

  Specifies that the logical area is a system record that is used to identify hash buckets. Users cannot explicitly create these types of logical areas. This type should *not* be used for the RDB$SYSTEM logical areas. This type does not identify system relations.

- Type=Blob

  Specifies that the logical area is a BLOB (LIST OF BYTE VARYING) repository.

There is no error checking of the type specified for a logical area. The specified type does not affect the collection of statistics, nor does it affect the readying of the affected logical areas. However, an incorrect type will cause incorrect statistics to be reported by the RMU Show Statistics utility.

Usage Notes

No new usage notes.

Examples

Example 1 Updating AIP Type for a Table

```
$ RMU/REPAIR MF_PERSONNEL /INITIALIZE=LAREA_PARAMETERS=AREAS.OPT
$ Type the areas.opt file to show the contents of the file.
$ TYPE AREAS.OPT
EMPLOYEES /TYPE=TABLE
```

Example 2 Updating AIP Type for a Storage Area

```
$ RMU/REPAIR MF_PERSONNEL /INITIALIZE=LAREA_PARAMETERS=AREAS.OPT
$ Type the areas.opt file to show the contents of the file.
$ TYPE AREAS.OPT
EMPLOYEES /AREA=EMPIDS_OVER /TYPE=TABLE
```

RMU Restore Command

Restores a database to the condition it was in at the time a full or incremental backup operation was performed with an RMU Backup command. In addition, if after-image journal (.aij) files have been retained, Oracle RMU will attempt to apply any preexisting .aij files to recover the database completely. See the Oracle Rdb installation information for details on the conditions under which Oracle RMU attempts an automatic .aij file recovery as part of the restore operation.

When you use the RMU Restore command to restore the database to a system with a more recent version of Oracle Rdb software, an RMU Convert command with the Noconfirm and Commit qualifiers is automatically executed as part of the RMU Restore command. Therefore, by executing the RMU Restore command, you convert that database to the current version. See the Oracle Rdb installation information for the proper backup procedure prior to installing a new release of Oracle Rdb and restoring (or converting) databases.

When you restore a database, default or propagated OpenVMS access control entries (ACEs) for the database root (.rdb) file take precedence over any Oracle RMU database access you might have. Therefore, if default or propagated entries are in use, you must use the RMU Show Privilege and RMU Set Privilege commands after a restore operation completes to verify and correct the Oracle RMU access. (You can tell if default or propagated entries are in use because the RMU Restore command displays the warning message "RMU-W-PREVACL, Restoring the root ACL over a pre-existing ACL." This is a normal condition if Oracle RMU was invoked from the CDO utility.)

To use RMU Show Privilege and RMU Set Privilege commands, you must have the rights to edit the access control list (ACL) using RMU$SECURITY access (which is VMS BIT_15 access in the access control entry (ACE)) and also (READ+WRITE+CONTROL) access. (Note that you can grant yourself BIT_15 access by using the DCL SET ACL command if you have (READ+WRITE+CONTROL) access.

If you do not have the required access after a restore operation to make the needed changes, someone with the required access or OpenVMS BYPASS or SECURITY access must examine and correct the ACL.

This behavior exists in Oracle RMU to prevent someone from using Oracle RMU to override the existing OpenVMS security policy.

Format

RMU/Restore backup-file-spec [storage-area-name[,...]]

Command Qualifiers	Defaults
[No]Acl	Acl
Active_IO=max-reads	Active_IO=3
[No]After_journal=file-spec	See description
[No]Aij_Options=journal-opts	See description
Area	See description
[No]Cdd_Integrate	Cdd_Integrate
[No]Commit	Commit
[No]Confirm	See description
Directory=directory-spec	See description
Disk_File[=(Reader_Threads=n)]	See description
[No]Duplicate	Noduplicate
Global_Buffers=global-buffer-options	Current value
Incremental	Full restore
Journal=file-name	See description
Just_Corrupt	See description
Label=(label-name-list)	See description
Loader_Synchronization	See description
Local_Buffers=local-buffer-options	Current value
[No]Log	See description
Master	See description
[No]Media_Loader	See description
[No]New_Version	Nonew_Version
Nodes_Max=number-cluster-nodes	See description
[No]Online	Noonline
Open_Mode={Automatic | Manual}	Current value
Options=file-spec	None
Page_Buffers=number-buffers	Page_Buffers=3
Path=cdd-path	Existing value
Prompt={Automatic or Operator or Client}	See description
[No]Recovery[=Aij_Buffers=n]	See description
[No]Rewind	Norewind
Root=root-file-spec	Existing value
Transaction_Mode=(mode-list)	Transaction_Mode=Current
Users_Max=number-users	Existing value
Volumes=n	Volumes=1

File or Area Qualifiers	Defaults
Blocks_Per_Page=integer	See description
Extension={Disable | Enable}	Current value
File=file-spec	See description
Just Corrupt	See description
Read_Only	Current value
Read_Write	Current value
Snapshot=(Allocation=n,File=file-spec)	See description
[No]Spams	Current value
Thresholds=(val1[,val2[,val3]])	Current value
[No]Worm	Current value

Command Parameters

backup-file-spec

A file specification for the backup file produced by a previous RMU Backup command. Note that you cannot perform a remote restore operation on an .rbf file that has been backed up to tape and then copied to disk.

The default file extension is .rbf.

Depending on whether you are performing a restore operation from magnetic tape, disk, or multiple disks, the backup file specification should be specified as follows:

- If you are restoring from magnetic tape
  - If you used multiple tape drives to create the backup file, the backup-file-spec parameter must be provided with (and only with) the first tape drive name. Additional tape drive names must be separated from the first and subsequent tape drive names with commas, as shown in the following example:

```
$ RMU/RESTORE /REWIND $111$MUA0:PERS_FULL_NOV30.RBF,$112$MUA1:
```

- If you are restoring from multiple or single disk files
  - If you used multiple disk files to create the backup file, the backup-file-spec parameter must be provided with (and only with) the first disk device name. Additional disk device names must be separated from the first and subsequent disk device names with commas. You must also be sure to include the Disk_File qualifier. For example:

```
$ RMU/RESTORE/DISK_FILE DISK1:[DIR1]MFP.RBF,DISK2:[DIR2],DISK3:[DIR3]
```

As an alternative to listing the disk device names on the command line (which, if you use several devices, can exceed the line-limit length for a command line), you can specify an options file in place of the backup-file-spec. For example:

```
$ RMU/RESTORE/DISK_FILE "@DEVICES.OPT"
```

The contents of devices.opt might appear as follows:

```
DISK1:[DIR1]MFP.RBF
DISK2:[DIR2]
DISK3:[DIR3]
```

The backup files referenced from such an options file are:

```
DISK1:[DIR1]MFP.RBF
DISK2:[DIR2]MFP01.RBF
DISK3:[DIR3]MFP02.RBF
```

Command Qualifiers

Commit
Nocommit

This qualifier is only meaningful when the backup file being restored is from a previous version of Oracle Rdb.

The Commit qualifier instructs Oracle RMU to commit the converted database to the current version of Oracle Rdb before completing the restore operation. In this case, the conversion is permanent and the database cannot be returned to the previous version. The Nocommit qualifier instructs Oracle RMU not to commit the converted database. In this case, you can roll back the database to its original version using the RMU Convert command with the Rollback qualifier, or you can permanently commit it to the current version by issuing the RMU Convert command with the Commit qualifier.

The Commit qualifier is the default.

Disk_File=[(Reader_Threads=integer)]

Specifies that you want to perform a multithreaded restore operation from disk files, floppy disks, or other disks external to the PC. This qualifier must have been specified on the RMU Backup command when the backup files from which you are restoring were created.

The Reader_Threads keyword specifies the number of threads that Oracle RMU should use when performing a multithreaded restore operation from disk files. You can specify no more than one reader thread per device specified on the command line (or in the command parameter options file). By default, one reader thread is used.

This qualifier and all qualifiers that control tape operations (Label, Loader_Synchronization, Master, Media_Loader, and Rewind) are mutually exclusive.

Prompt=Automatic
Prompt=Operator
Prompt=Client

Specifies where server prompts are to be sent. When you specify Prompt=Automatic, prompts are sent to the standard input device, and when you specify Prompt=Operator, prompts are sent to the server console. When you specify Prompt=Client, prompts are sent to the client system.

Transaction_Mode=(modes-list)

Sets the allowable transaction modes for the database root file created by the restore operation. The modes-list can include one or more of the following transaction modes:

- All - Enables all transaction modes.
- Current - Enables all transaction modes that are set for the source database. This is the default transaction mode.
- None - Disables all transaction modes.
- [No]Batch_Update
- [No]Read_Only
- [No]Exclusive
- [No]Exclusive_Read
- [No]Exclusive_Write
- [No]Protected
- [No]Protected_Read
- [No]Protected_Write
- [No]Read_Write
- [No]Shared
- [No]Shared_Read

- [No]Shared_Write

Your restore operation must include the database root file. Otherwise, RMU returns the CONFLSWIT error when you issue an RMU Restore command with the Transaction_Mode qualifier.

If you specify more than one transaction mode in the modes-list, enclose the list in parentheses and separate the transaction modes from one another with a comma. Note the following:

- When you specify a negated transaction mode, it indicates that a mode is not an allowable access mode. For example, if you specify the Noexclusive_Write access mode, it indicates that exclusive write is not an allowable access mode for the restored database.
- If you specify the Shared, Exclusive, or Protected transaction mode, Oracle RMU assumes that you are referring to both reading and writing in that transaction mode.
- No mode is enabled unless you add that mode to the list, or you use the All option to enable all transaction modes.
- You can list one transaction mode that enables or disables a particular mode followed by another that does the opposite.

  For example, Transaction_Mode=(Noshared_Write, Shared) is ambiguous because the first value disables Shared_Write access and the second value enables Shared_Write access. Oracle RMU resolves the ambiguity by first enabling the modes as specified in the modes-list and then disabling the modes as specified in the modes-list. The order of items in the list is irrelevant. In the example presented previously, Shared_Read is enabled and Shared_Write is disabled.

Volumes=n

Specifies that concurrent tape or disk access is to be used to accelerate the restore operation.

If you are restoring from tape, the value of "n" indicates the number of tape volumes containing the backup file. The number of volumes must be specified accurately for the restore operation to complete.

If you are restoring from a multidisk backup file, the value of "n" indicates the number of disk devices containing backup files needed for the restore operation.

Usage Notes

No new usage notes.

Examples

Example 1 Setting the Transaction Mode to Read-Only for a Restored Database

```
RMU/RESTORE MF_PERSONNEL.RBF/TRANSACTION_MODE=READ_ONLY/NOCDD/NORECOVER
%RMU-I-AIJRSTAVL, 3 after-image journals available for use
%RMU-I-AIJRSTMOD, 2 after-image journals marked as "modified"
%RMU-I-AIJISON, after-image journaling has been enabled
%RMU-W-DOFULLBCK, full database backup should be done to ensure future recovery
$ SQL
SQL> ATTACH 'FILENAME MF_PERSONNEL.RDB';
SQL> INSERT INTO DEPARTMENTS
cont> (DEPARTMENT_NAME, BUDGET_ACTUAL)
cont> VALUES
cont> ('RECR',
cont>  128776);
%RDB-E-READ_ONLY_TRANS, attempt to update during a read-only transaction
```

Example 2 Restoring a Database and Specifying Nocommit

```
$ ! Restore a backup file of a database that has a structure level of
$ ! Rdb7 in an V7.1 environment
$ RMU/SHOW VERSION
Executing RMU for Oracle Rdb V7.1-00
$ RMU/RESTORE MFP70.RBF /NOCOMMIT/NOCDD/NORECOVER
%RMU-I-AIJRSTAVL, 0 after-image journals available for use
%RMU-I-AIJISOFF, after-image journaling has been disabled
%RMU-I-LOGCONVRT, database root converted to current structure level
%RMU-S-CVTDBSUC, database USER1:[80]MF_PERSONNEL.RDB;1 successfull
y converted from version V7.0 to V7.1
%RMU-W-USERECCOM, Use the RMU Recover command. The journals are not available.
$ RMU/SHOW VERSION
Executing RMU for Oracle Rdb V7.1-00
$ RMU/CONVERT/ROLLBACK MF_PERSONNEL.RDB
%RMU-I-RMUTXT_000, Executing RMU for Oracle Rdb X7.1-00
Are you satisfied with your backup of RDBVMS_USER1:[V70]MF_PERSONNEL.RDB;
1 and your backup of any associated .aij files [N]? Y
%RMU-I-LOGCONVRT, database root converted to current structure level
%RMU-I-CVTROLSUC, CONVERT rolled-back for RDBVMS_USER1:[V70]MF_PERSONNEL.
RDB;1 to version V7.0
```

RMU Restore Only_Root Command

Permits you to recover more quickly from the loss of a database root (.rdb) file by restoring only the root file. This command is not valid for single-file databases.

Format

RMU/Restore/Only_Root backup-file-spec [storage-area-list]

Command Qualifiers	Defaults
Active_IO=max-reads	Active_IO=3
[No]After_Journal=file-spec	See description
No]Aij_Options=journal-opts	See description
Directory=directory-spec	See description
No]Initialize_Tsns	Noinitialize_Tsns
Label=(label-name-list)	See description
[No]Log	See description
[No]Media_Loader	See description
[No]New_Snapshots	Nonew_Snapshots
Nodes_Max=number-cluster-nodes	Existing value
Options=file-spec	None
[No]Rewind	Norewind
Root=root-file-spec	Existing value
[No]Set_Tsn=(Tsn=n,Csn=m)	See description
Transaction_Mode=(modes-list)	Transaction_Mode=Current
[No]Update_Files	Update_Files
Users_Max=numbers-users	Existing value

File or Area Qualifiers	Defaults
[No]Blocks_Per_Page=integer	Noblocks_Per_Page
File=file-spec	See description
Read_Only	Current value
Read_Write	Current value
Snapshot=(Allocation=n,File=file-spec)	See description
[No]Spams	Current value
Thresholds=(val1[,val2[,val3]])	Current value
[No]Worm	Current value

Command Parameters

backup-file-spec

A file specification for the backup file produced by a previous RMU Backup command. Note that you cannot perform a remote restore operation on an .rbf file that has been backed up to tape and then copied to disk.

The default file extension is .rbf.

Depending on whether you are performing a restore operation from magnetic tape, disk, or multiple disks, the backup file specification should be specified as follows:

- If you are restoring from magnetic tape

  If you used multiple tape drives to create the backup file, the backup-file-spec parameter must be provided with (and only with) the first tape drive name. Additional tape drive names must be separated from the first and subsequent tape drive names with commas, as shown in the following example:

  ```
  $ RMU/RESTORE /REWIND $111$MUA0:PERS_FULL_NOV30.RBF,$112$MUA1:
  ```

- If you are restoring from multiple or single disk files

  If you used multiple disk files to create the backup file, the backup-file-spec parameter must be provided with (and only with) the first disk device name. Additional disk device names must be separated from the first and subsequent disk device names with commas. You must also be sure to include the Disk_File qualifier. For example:

  ```
  $ RMU/RESTORE/ONLY_ROOT/DISK_FILE DISK1:[DIR1]MFP.RBF,DISK2:[DIR2],
  DISK3:[DIR3]
  ```

  As an alternative to listing the disk device names on the command line (which, if you use several devices, can exceed the line-limit length for a command line), you can specify an options file in place of the backup-file-spec. For example:

  ```
  RMU/RESTORE/ONLY_ROOT/DISK_FILE" @DEVICES.OPT"
  ```

  The contents of devices.opt might appear as follows:

  ```
  DISK1:[DIR1]MFP.RBF
  DISK2:[DIR2]
  DISK3:[DIR3]
  ```

  The backup files referenced from such an options file are:

  ```
  DISK1:[DIR1]MFP.RBF
  ```

```
DISK2:[DIR2]MFP01.RBF
DISK3:[DIR3]MFP02.RBF
```

Command Qualifiers

Transaction_Mode=(modes-list)

Sets the allowable transaction modes for the database root file created by the restore operation. The modes-list can include one or more of the following transaction modes:

- All - Enables all transaction modes.
- Current - Enables all transaction modes that are set for the source database. This is the default transaction mode.
- None - Disables all transaction modes.
- [No]Batch_Update
- [No]Read_Only
- [No]Exclusive
- [No]Exclusive_Read
- [No]Exclusive_Write
- [No]Protected
- [No]Protected_Read
- [No]Protected_Write
- [No]Read_Write
- [No]Shared
- [No]Shared_Read
- [No]Shared_Write

If you specify more than one transaction mode in the modes-list, enclose the list in parentheses and separate the transaction modes from one another with a comma. Note the following:

- When you specify a negated transaction mode, it indicates that a mode is not an allowable access mode. For example, if you specify the Noexclusive_Write access mode, it indicates that exclusive write is not an allowable access mode for the restored database.

- If you specify the Shared, Exclusive, or Protected transaction mode, Oracle RMU assumes that you are referring to both reading and writing in that transaction mode.
- No mode is enabled unless you add that mode to the list, or you use the All option to enable all transaction modes.
- You can list one transaction mode that enables or disables a particular mode followed by another that does the opposite.

  For example, Transaction_Mode=(Noshared_Write, Shared) is ambiguous because the first value disables Shared_Write access and the second value enables Shared_Write access. Oracle RMU resolves the ambiguity by first enabling the modes as specified in the modes-list and then disabling the modes as specified in the modes-list. The order of items in the list is irrelevant. In the example presented previously, Shared_Read is enabled and Shared_Write is disabled.

Usage Notes

No new usage notes.

RMU Set Privilege Command

Allows you to modify the root file access control list (ACL) for a database.

A database root file ACL determines which Oracle RMU commands users can execute for the associated database.

Format

RMU/Set Privilege root-file-spec

Command Qualifiers	Defaults
Acl[=(ace[,,,])	See description
Acl_File=filename	See description
After=ace	See description
Delete[All]	See description
Edit	No editor invoked
[No]Journal[=file-spec]	Journal
Keep[=(Recovery_Journal)]	See description
Like=source-root-file-spec	None
[No]Log	Nolog
Mode=[No]Prompt	Mode=Prompt
New	None
[No]Recover[=file-spec]	Norecover
Replace=(ace[,...])	None

Command Qualifiers

Acl
Acl[=(ace[,...])]
Specifies one or more access control entries (ACEs) to be modified. When no ACE is specified, the entire ACL is affected. Separate multiple ACEs with commas. You cannot specify the Acl qualifier and the Acl_File qualifier on the same RMU command line.

The format of an ACE is as follows:

(Identifier=user-id, Access=access_mask)

The user-id must be one of the following types of identifier:

- A user identification code (UIC) in [group-name,member-name] alphanumeric format

- A user identification code (UIC) in [group-number,member-number] numeric format
- A general identifier, such as SECRETARIES
- A system-defined identifier, such as DIALUP
- Wildcard characters in [*,*] format

Names are not case sensitive. In addition, the Identifier and Access keywords can be abbreviated to one character. For example, the following ACE is valid:

(I=isteward, A=RMU$ALL)

The access_mask may be any of the following:

- One or more of the Oracle RMU privileges listed in the *Oracle Rdb7 Oracle RMU Reference Manual*

  If more than one privilege is specified, a plus sign (+) must be placed between each privilege.
- The keyword RMU$ALL

  These keywords indicate that you want the user to have all of the RMU privileges. (This keyword has no effect on system file privileges.)
- The keyword None

  This keyword indicates that you do not want the user to have any RMU or OpenVMS privileges. If you specify Acl=(id=username, access=READ+NONE), the specified user will have no RMU privileges and no READ privileges for the database files.

Acl_File=filename

Specifies a file containing a list of ACEs, with one ACE specified per line. You can use continuation characters to continue an ACE on the next line, and you can include commented lines within the file. Within this file, use the dash (-) as a continuation character and the exclamation point (!) to indicate a comment.

You cannot specify the Acl_File qualifier and the Acl qualifier on the same RMU command line.

Log
Nolog
The Log qualifier directs the RMU Set Privilege command to return the name of the root file that has been modified by the command and the ACL. The Nolog qualifier suppresses this output.

Replace
Replace=(ace[,...])
Deletes the ACEs specified with the Acl qualifier and replaces them with those specified with the Replace qualifier. Any ACEs specified with the Acl qualifier must exist and must be specified in the order in which they appear in the ACL.

This qualifier cannot be used with the Edit qualifier.

Usage Notes

- On OpenVMS, the RMU Set Privilege command with the ACL qualifier adds the specified ACEs at the top of the ACL unless you specify the After qualifier.
- When you issue the RMU Set Privilege command with the Log qualifier, Oracle RMU displays the ACL associated with the database.

RMU Show Privilege Command

Allows you to display the root file access control list (ACL) for a database.

Format

RMU/Show Privilege root-file-spec

Command Qualifiers	Defaults
[No]Expand_All	Noexpand_All
[No]Header	Header

Command Qualifiers

Expand_All
Noexpand_All
The Expand_All qualifier specifies that if a user's access mask was defined with the RMU$ALL keyword on the RMU Set Privilege command, then the RMU Show Privilege command will display each of the RMU privileges represented by the RMU$ALL keyword.

The Noexpand_All qualifier specifies that if a user's access mask was defined with the RMU$ALL keyword on the RMU Set Privilege command, then the RMU Show Privilege command will display the keyword, not the RMU privileges represented by the keyword.

The Noexpand_All qualifier is the default.

Header
Noheader
The Header qualifier specifies that header information will be displayed. The Noheader qualifier suppresses the output of the header information. The Header qualifier is the default.

Usage Notes

No new usage notes.

Examples

Example 1 Using the Header and Noheader Qualifiers

```
$! The following examples show the difference
$! in output when you use the Header and Noheader qualifiers:
$ RMU/SHOW PRIV MF_PERSONNEL.RDB/HEADER
Object type: file,  Object name: RDBVMS_USER:[DB]MF_PERSONNEL.RDB;1,  on
 17-SEP-1998 13:47:20.21
          (IDENTIFIER=[RDB,STONE],ACCESS=RMU$ALL)
$ RMU/SHOW PRIVILEGE MF_PERSONNEL.RDB/NOHEADER
          (IDENTIFIER=[RDB,STONE],ACCESS=RMU$ALL)
```

Example 2 Using the Expand and Noexpand Qualifiers

```
$ RMU/SET PRIVILEGE MF_PERSONNEL.RDB /ACL=(I=STONE,A=RMU$ALL)
$ RMU/SHOW PRIVILEGE MF_PERSONNEL.RDB /NOEXPAND/NOHEADER
(IDENTIFIER=[RDB,STONE],ACCESS=READ+WRITE+CONTROL+RMU$ALL)
$ RMU/SHOW PRIVILEGE MF_PERSONNEL.RDB /EXPAND/NOHEADER
(IDENTIFIER=[RDB,STONE],ACCESS=READ+WRITE+CONTROL+RMU$ALTER+
          RMU$ANALYZE+RMU$BACKUP+RMU$CONVERT+RMU$COPY+RMU$DUMP+RMU$LOAD+
          RMU$MOVE+RMU$OPEN+RMU$RESTORE+RMU$SECURITY+RMU$SHOW+RMU$UNLOAD+
          RMU$VERIFY)
```

RMU Show Statistics Command

Opens the Performance Monitor to display, on a character-cell terminal, the usage statistics for a database. See the *Oracle Rdb7 Guide to Database Performance and Tuning* for tutorial information on how to interpret the Performance Monitor displays available prior to this release. See Chapter 6 for information on how to interpret the Performance Monitor displays added to this release.

A windowing interface for the Performance Monitor is available. See the *Oracle Rdb Release Notes* for a list of supported PC clients.

Format

RMU/Show Statistics [root-file-spec]

Command Qualifiers	Defaults
Alarm=Interval	Alarm=0
[No]Broadcast	See description
[No]Cluster=[(node-list)]	Nocluster
Configure=file-spec	None
[No]Cycle=Seconds	Nocycle
Dbkey_Log=file-spec	See description
Deadlock_Log=file-spec	None
[No]Histogram	Histogram
Input=file-name	See description
[No]Interactive	See description
Lock_Timeout_Log=file-spec	None
[No]Log	See description
[No]Logical_Area	Logical_Area
No]Notify[=([No]All or operator classes)]	Nonotify
Options=options	Options=Base
Output=file-spec	See description
[No]Prompt_Timeout=seconds	Prompt_Timeout=60
Reopen_Interval=minutes	See description
Reset	See description
Screen=screen-name	See description
Stall_Log=file-spec	See description
Time=integer	Time=3
Until="date-time"	See description

Command Qualifiers

Cluster=(node-list)
Nocluster
Specifies the list of remote nodes from which statistics collection and presentation is to be performed. The collected statistics are merged with the information for the current node and displayed using the usual statistics screens.

The following list shows the use of the Cluster qualifier:

- If the Cluster qualifier is specified by itself, then remote statistics collection is performed on all cluster nodes on which the database is currently open.
- If the Cluster=(node-list) qualifier is specified, then remote statistics collection is performed on the specified nodes only, even if the database is not yet open on those nodes.
- If the Cluster qualifier is not specified, or the Nocluster qualifier (the default) is specified, then cluster statistics collection is not performed. However, you can still enable clusterwide statistics collection online using the Tools menu.

You can specify up to 95 different cluster nodes with the Cluster qualifier. There is a maximum number of 95 cluster nodes because Oracle Rdb only supports 96 nodes per database. The current node is always included in the list of nodes from which statistics collection is to be performed.

It is not necessary to have the RMU Show Statistics command running on the specified remote nodes or to have the database open on the remote nodes. These events are automatically handled by the feature.

The following example shows the use of the Cluster qualifier to initiate statistics collection and presentation from two remote nodes:

```
$ RMU /SHOW STATISTICS /CLUSTER=(BONZAI, ALPHA4) MF_PERSONNEL
```

Remote nodes can also be added and removed online at run time. Use the Cluster Statistics option located in the Tools menu. The Tools menu is displayed by using the exclamation point (!) on-screen menu option.

See Section 6.7 for information about the options on the Cluster Statistics menu.

Configure=file-spec
Specifies a human-readable configuration file. The configuration file can be created using any editor, or can be automatically generated from the RMU Show Statistics command using the current run-time configuration settings.

The configuration file is processed by the RMU Show Statistics command prior to opening the database or the binary input file, if you specify the Configure=file-spec qualifier. If you do not specify this qualifier, all of the variables are the defaults based on command-line qualifiers and logical names. The default configuration file type is .cfg. The configuration file is processed in two passes. The first pass occurs before the database is opened and processes most of the configuration file entries. The second pass occurs after the database is opened and processes those variables that are database-dependent, such as the CUSTOMER_LINE_n variable.

See Section 6.6 for more information and Section 6.6.10 for an example of a configuration file.

Deadlock_Log=file-spec

Records the last deadlock for the processes. There is no method to record each lock deadlock as it occurs.

The file-spec is the name of the file to which you want all lock deadlock messages to be logged. The lock deadlock messages are written in human-readable format similar to the Lock Timeout History and Lock Deadlock History screens. The header region of the lock deadlock log contains three lines. The first line indicates that the RMU Show Statistics utility created the log file. The second line identifies the database. The third line identifies the date and time the log was created.

The main body of the stall log contains three columns:

- The first column contains the process ID and stream ID that experienced the lock deadlock.
- The second column contains the time the deadlock occurred; however, the date is not displayed.
- The third column contains the deadlock message describing the affected resource; this message is similar to the originating stall message.

For example:

```
2EA00B52:34 14:25:46.14 - waiting for page 5:751 (PR)
```

If any lock deadlocks are missed for a particular process (usually because the recording interval is too large), then the number of missed lock deadlocks is displayed in brackets after the message. For example:

```
2EA00B52:34 14:25:46.14 - waiting for page 5:751 (PR) [1 missed]
```

Only one message per occurring deadlock is logged.

The lock deadlock messages are written at the specified screen refresh rate, determined by specifying the Time qualifier, or online using the Set_rate on-screen menu option. Using a larger refresh rate minimizes the size of the file, but results in a large number of missed deadlock messages. Using a smaller refresh rate produces a large log file, but contains a much finer granularity of deadlock messages.

The affected LockID is not displayed, because this is meaningless information after the lock deadlock has completed.

Using the Time=1 or Time=50 qualifier produces a reasonable log while minimizing the impact on the system.

Use the Tools menu (displayed when you press the exclamation point (!) key from any screen) to enable or disable the lock timeout and lock deadlock logging facility while the RMU Show Statistics utility is running. However, note that the lock timeout log and lock deadlock log are not available during binary file replay.

Lock_Timeout_Log=file-spec

Records the last lock timeout message for the processes. There is no method to record each lock timeout as it occurs. The lock timeout messages are written in human-readable format. The header region of the lock timeout log contains three lines. The first line indicates that the RMU Show Statistics utility created the log file. The second line identifies the database. The third line identifies the date and time the log was created.

The main body of the stall log contains three columns:

- The first columns contains the process ID and stream ID that experienced the lock timeout.
- The second column contains the time the timeout occurred; however, the date is not displayed.
- The third column contains the timeout message describing the affected resource; this message is similar to the originating stall message.

For example:

```
2EA00B52:34 14:25:46.14 - waiting for page 5:751 (PR)
```

If any lock timeouts are missed for a particular process (usually because the recording interval is too large), then the number of missed lock timeouts is displayed in brackets after the message. For example:

```
2EA00B52:34 14:25:46.14 - waiting for page 5:751 (PR) [1 missed]
```

Only one message per occurring lock timeout is logged.

The lock timeout messages are written at the specified screen refresh rate, determined by specifying the Time qualifier, or online using the Set_rate on-screen menu option. Using a larger refresh rate minimizes the size of the file, but results in a large number of missed lock timeout messages. Using a smaller refresh rate produces a large log file, but contains a much finer granularity of lock timeout messages.

The affected LockID is not displayed, because this is meaningless information after the lock timeout has completed.

Note that you do not need to be displaying the Lock Timeout History or Lock Deadlock History screens to record the stall messages to the stall log. These logs are maintained regardless of which screen, if any, is displayed.

Using the Time=1 or Time=50 qualifier appears to produce a reasonable log while minimizing the impact on the system.

Use the Tools menu (displayed when you press the exclamation point (!) key from any screen) to enable or disable the lock timeout and lock deadlock logging facility while the RMU Show Statistics utility is running. However, note that the lock timeout log and lock deadlock log are not available during binary file replay.

Logical_Area
Nological_Area

The Logical_Area qualifier specifies that you want the RMU Show Statistics command to acquire the needed amounts of virtual memory to display logical area statistics information.

By default, the RMU Statistics command consumes approximately 13,000 bytes of virtual memory per logical area. (The number of logical areas is determined by the largest logical area identifier, not the actual number of areas.)

This can result in the RMU Show Statistics command consuming large amounts of virtual memory, even if you do not want to review logical area statistics information.

Use the Nological_Area qualifier to indicate that you do not want to display logical area statistics information. When you specify the Nological_Area qualifier, the virtual memory for logical area statistics information presentation is not acquired.

Be careful when you specify the Nological_Area qualifier that you do not also specify the Nolog qualifier, as this causes logical area statistics information to still be collected.

There is no corresponding configuration variable. This qualifier cannot be modified at run time. See Section 6.2 for information on interpreting the screens produced by the Logical_Area qualifier.

The Logical_Area qualifier is the default.

Option=Confirm
The Confirm keyword indicates that you wish to confirm before exiting from the utility.

You can also specify this qualifier in the configuration file using the CONFIRM_EXIT variable. A value of TRUE indicates that you want to confirm before exiting the utility and a value of FALSE, the default value, indicates you do not want to confirm before exiting the utility.

Option=Verbose
Causes the stall message logging facility to report a stall message at each interval, even if the stall message has been previously reported.

Note: Use of the Option=Verbose qualifier can result in an enormous stall messages log file. Ensure that adequate disk space exists for the log file when you use this qualifier.

You can enable or disable the stall messages logging verbose option at run time by using the Tools menu and pressing the exclamation point (!) key.

You can also specify the verbose option in the configuration file, using the STALL_LOG_VERBOSE variable. Valid keywords are ENABLED or DISABLED.

Lock information is displayed only once per stall, even in verbose mode, to minimize the output file size. See Section 6.1.4 for an example.

Prompt_Timeout=seconds
Noprompt_Timeout
Allows you to specify the user prompt timeout interval, in seconds. The default value is 60 seconds.

If you specify the Noprompt_Timeout qualifier or the Prompt_Timeout=0, then the RMU Show Statistics command does not timeout any user prompts. Note that this can cause your database to hang.

Note: Oracle Corporation recommends that you do not use the Noprompt_Timeout qualifier or the Prompt_Timeout= 0 qualifier unless you are certain that prompts will always be responded to in a timely manner.

If the Prompt_Timeout qualifier is specified with a value greater than 0 but less than 10 seconds, the value 10 is used. The user prompt timeout interval can also be specified using the PROMPT_TIMEOUT configuration variable.

Usage Notes

- Previously, the Oracle Rdb RMU Show Statistics command was unable to correctly display process CPU times in excess of 1 day; the number of days value was not displayed.

  The RMU Show Statistics command is now able to display CPU times greater than 1 day. Because the width of the CPU time display is limited, the following CPU time display formats are used:

  – For CPU time values less than 1 day: "HH:MM:SS.CC"

  – For CPU time values less than 100 days but more than 1 day: "DDHH:MM"

  – For CPU time values more than 100 days: "DDD HH:MM"

- Virtual memory statistics are no longer collected or displayed.

  Oracle Rdb no longer collects virtual memory (VM) statistics and the information is no longer included in the output from the RMU Show Statistics command. This information includes the following RMU Show Statistics items:

  – GET_VM calls

  – FREE_VM calls

  – GET_VM kilobytes

  – FREE_VM kilobytes

  – $EXPREG calls

- The following caveats apply to the Cluster Statistics Collection and Presentation feature:

  – Up to 95 cluster nodes can be specified. However, use cluster statistics collection prudently, as the system overhead in collecting the remote statistics may be substantial depending on the amount of information being transmitted on the network.

– Cluster statistics are collected at the specified display refresh rate. Therefore, set the display refresh rate to a reasonable rate based on the number of cluster nodes being collected. The default refresh rate of 3 seconds is reasonable for most remote collection loads.

– If you specify the Cluster qualifier, the list of cluster nodes applies to any database accessed during the Show Statistics session. When you access additional databases using the Switch Database option, the same cluster nodes are automatically accessed. However, any nodes that you added manually using the Cluster Statistics menu are not automatically added to the new database's remote collection.

  In other words, manually adding and deleting cluster nodes affects only the current database and does not apply to any other database that you may have accessed during the session. For example, when you run the Show Statistics utility on node ALPHA3 with manually added node BONZAI, subsequently switching to BONZAI as the current node will not display cluster statistics from node ALPHA3 unless you manually add that node. Furthermore, switching back to node ALPHA3 as the current node loses the previous collection of node BONZAI because it was manually added.

– Both DECnet and TCP/IP network protocols are supported. By default, the DECnet protocol is used. To explicitly specify which network protocol to use, define the RDM$BIND_STT_NETWORK_TRANSPORT to DECNET or TCPIP respectively. The RDM$BIND_STT_NETWORK_TRANSPORT logical name must be defined to the same definition on both the local and cluster nodes. The RDM$BIND_STT_NETWORK_TRANSPORT logical name can be specified in LNM$FILE_DEV on the local node but must be specified in the LNM$SYSTEM_TABLE on all remote nodes.

  Note: There is no command qualifier to specify the network protocol.

– The Output qualifier continues to work as usual, but when in cluster mode writes the cluster statistics information to the binary output file.

– The Cluster qualifier cannot be specified with the Input qualifier. Furthermore, the online selection of cluster nodes is not available when you use the Input qualifier.

- While the collection and presentation feature is active, all on-screen menu options continue to operate as usual. This includes the time-plot, scatter-plot, screen pause, and various other options.
- There is no way to exclude the current node from statistics collection. Log in to another node if you want to do this.
- The cluster collection of per-process stall information automatically detects the binding or unbinding of processes to cluster databases. There is no need to manually refresh the database information on the current node.
- If the database is not currently open on the specified node, Oracle RMU still attempts to collect cluster statistics. However, you must open the remote database prior to regular process attaches.
- When you display any of the per-process screens that support cluster statistics collection, such as the Stall Messages screen, you can zoom in on any of the displayed processes to show which node that process is using.
- Using the Cluster Statistics submenu from the Tools menu, it is also possible to collect statistics from all open database nodes using the Collect From Open Database Nodes menu option. This option simplifies the DBA's job of remembering where the database is currently open. However, subsequently opened nodes are not automatically added to the collection; these must be manually added.
- The cluster statistics collection is an *intracluster* feature in that it only works on the same database, using the same device and directory specification used to run the initial RMU Show Statistics command (that is, on a shared disk). The cluster statistics collection does not work across clusters (*intercluster*).
- When you replay a binary output file, the screen header region accurately reflects the number of cluster nodes whose statistics are represented in the output file.

Examples

Example 1 Creating a Lock Event Logging Server

Using the Lock_Timeout or Deadlock qualifiers, you can construct a Lock Event Logging server. The following OpenVMS DCL script shows how to create a server that logs both lock timeout and lock deadlock events on the mf_personnel database for the next 15 minutes:

```
$ RMU/SHOW STATISTICS /NOHISTOGRAM /TIME=1 /NOINTERACTIVE -
_$ /LOCK_TIMEOUT_LOG=TIMEOUT.LOG /DEADLOCK_LOG=DEADLOCK.LOG -
_$ /NOBROADCAST /UNTIL="+15:00" MF_PERSONNEL
```

RMU Verify Command

Checks the internal integrity of database data structures. The RMU Verify command does not verify the data itself. You can verify specific portions of a database or the integrity of routines stored in the database by using qualifiers.

Format

RMU/Verify root-file-spec

Command Qualifiers	Defaults
All	See description
Areas[=storage-area-list]	No area checking performed
Checksum_Only	Full page verification
[No]Constraints=[(options)]	No constraint evaluation
[No]Data	Data when Indexes is used
End=page-number	End=last-page
[No]Functions	Nofunctions
Incremental	See description
Indexes=index-list	No index checking performed
Lareas[=logical-area-list]	No LAREA checking performed
[No]Log	See description
Output=file-spec	See description
[No]Root	Root
[No]Routines	Noroutines
[No]Segmented_Strings	See description
Snapshots	No snapshot verification
Start=page-number	Start=1
Transaction_Type=option	Transaction_Type=Protected

Usage Notes

The RMU Verify command ignores any constraint that has been disabled (with the SQL ALTER TABLE enable-disable clause) unless you specify the constraint name in the Constraints=(Constraint=list) qualifier of the RMU Verify command. If the Constraints qualifier is specified without a list, then disabled constraints are ignored.

By specifying the name of a disabled constraint in the Constraints=(Constraint=list) qualifier, you can check it periodically without having to reenable it. You might use this to provide a business rule in the database that needs checking only occasionally.

This is a useful practice if the overhead of checking the constraint during operating hours is too expensive, or if it is already being enforced by the application.

Examples

Example 1 Verify a Database with Disabled Constraints

```
$ ! The following example demonstrates that the
$ ! RMU Verify command verifies disabled
$ ! constraints only when you explicitly specify
$ ! the disabled constraint.
$ SQL
SQL> ATTACH 'FILENAME MF_PERSONNEL.RDB';
SQL> -- Disable the EMP_SEX_VALUES constraint.
SQL> ALTER TABLE EMPLOYEES DISABLE CONSTRAINT EMP_SEX_VALUES;
SQL> COMMIT;
SQL> -- Insert a value that violates the EMP_SEX_VALUES constraint.
SQL> INSERT INTO EMPLOYEES
cont> (EMPLOYEE_ID, LAST_NAME, SEX)
cont> VALUES ('99999', 'JICKLING', 'G');
1 row inserted
SQL> COMMIT;
SQL> EXIT;
$ ! The following two verify commands do not return an error
$ ! because the disabled constraint is not explicitly specified.
$ RMU/VERIFY MF_PERSONNEL.RDB
$ RMU/VERIFY MF_PERSONNEL.RDB/CONSTRAINTS
$ ! The following verify command returns an informational message to
$ ! inform you that data that violates the disabled constraint
$ ! has been inserted into the database.
$ RMU/VERIFY MF_PERSONNEL.RDB/CONSTRAINT=(CONSTRAINT=EMP_SEX_VALUES)
%RMU-I-CONSTFAIL, Verification of constraint "EMP_SEX_VALUES" has failed.
```

6

Using the New RMU Show Statistics Screens

This chapter describes how to use the new screens generated by the RMU Show Statistics command.

6.1 Stall Class Informational Screens

The existing RMU Show Statistics Stall Messages and Active User Stall Messages screens are useful for identifying the cause of present stalls when you are diagnosing performance problems. However, these screens show only the current view of the cause of the stalls. Two new screens have been added to the RMU Show Statistics utility; each of these screens displays information about the types or classes of stalls that are occurring currently or have occurred since statistics collection began. There are 10 classes of stall messages:

- Records - record-related stalls, such as waiting for record locks to be granted
- Pages - page-related stalls, such as waiting for storage area I/O to complete or page locks to be granted
- Tables - table-related stalls, such as waiting for logical area locks to be granted
- Storage areas - storage-area-related stalls, such as waiting for storage areas to be created, deleted, truncated, or opened
- Database root file - database root-file-related stalls, such as waiting for root file I/O to complete or object locks to be granted
- Recovery journals - journal-related stalls, such as opening, initializing, or extending journals, as well as waiting for journal locks to be granted

- Transactions - transaction-related stalls, such as waiting for two-phase commit transactions to commit or waiting for checkpoints to complete
- Hot standby - hot-standby-related stalls
- Database - database-related stalls, such as waiting for the database freeze to complete
- Miscellaneous - generic stalls, such as waiting for a bugcheck dump to complete

The new Stall Statistics screen identifies the number of stalls and their corresponding durations for a particular stall class. The screen can be configured (using the Config on-screen menu option) to display either the aggregate count information or the aggregate duration information. The following example shows the Stall Statistics screen displaying aggregate count information.

```
Node: ALPHA3         Oracle Rdb V7.1    Performance Monitor 24-OCT-1996 15:55:30
Rate: 1.00 Second     Stall Statistics (Aggregate counts)   Elapsed: 01:44:29.27
Page: 1 of 1          USER1:[SPANDERSON.OE_MASTER]OE_RDB.RDB;1      Mode: Online
--------------------------------------------------------------------------------
statistic.........      rate.per.second............. total....... average......
name..............      max..... cur..... avg....... count....... per.trans....
miscellaneous                  0        0        0.0            0           0.0
records                       15        0        0.0          160           0.3
pages                       2706        0       48.1       301991         579.6
tables                         0        0        0.0            0           0.0
storage areas                 33        0        0.0          133           0.2
database rootfile            125        0        0.3         2162           4.1
recovery journals             66        0        0.5         3405           6.5
transactions                  69        0        0.0          294           0.5
hot standby                    0        0        0.0            0           0.0
database                       9        0        0.0           19           0.0
--------------------------------------------------------------------------------
Config Exit Graph Help Menu Options Pause Reset Set_rate Time_plot Unreset Write
```

This example shows there were 301,991 total stalls for the page class.

The following example shows the Stall Statistics screen displaying aggregate duration information:

```
Node: ALPHA3         Oracle Rdb V7.1    Performance Monitor 24-OCT-1996 15:55:41
Rate: 1.00 Second    Stall Statistics (Aggregate durations) Elapsed: 01:44:40.52
Page: 1 of 1          USER1:[SPANDERSON.OE_MASTER]OE_RDB.RDB;1      Mode: Online
--------------------------------------------------------------------------------
statistic.........      rate.per.second............. total....... average......
name..............      max..... cur..... avg....... count....... per.trans....
miscellaneous                  0        0        0.0            0           0.0
```

```
records                    8720        0       4.8       30237          58.0
pages                     78453        0      88.3      554787        1064.8
tables                        0        0       0.0           0           0.0
storage areas              3400        0       0.7        4742           9.1
database rootfile          1876        0       1.5        9835          18.8
recovery journals          6652        0       9.9       62716         120.3
transactions              31192        0      10.0       63145         121.1
hot standby                   0        0       0.0           0           0.0
database                   2052        0       0.1        1226           2.3
--------------------------------------------------------------------------------
Config Exit Graph Help Menu Options Pause Reset Set_rate Time_plot Unreset Write
```

The stall durations are collected as hundredths of seconds. Therefore, the total page stall duration of 554787 is 5547.87 seconds.

Note: Because certain types of stalls can be nested, the total stall durations may be reported as longer than actually occurred.

The new Active Stall Counts screen identifies the actual number of processes currently stalled in a particular stall class. Ideally, the number of stalled processes for each class should be 0, which indicates that there are no stalled processes.

The following example shows the Active Stall Counts screen:

```
Node: ALPHA3        Oracle Rdb V7.1    Performance Monitor 24-OCT-1996 15:41:05
Rate: 0.10 Seconds           Active Stall Counts          Elapsed: 01:30:04.69
Page: 1 of 1         USER1:[SPANDERSON.OE_MASTER]OE_RDB.RDB;1      Mode: Online
--------------------------------------------------------------------------------
Stall Category   Stall.Cnt       10   20   30   40   50   60   70   80   90  100
                             +----+----+----+----+----+----+----+----+----+----+
miscellaneous           0    |    |    |    |    |    |    |    |    |    |    |
records                 0    |    |    |    |    |    |    |    |    |    |    |
pages                   5    +-*  |    |    |    |    |    |    |    |    |    |
tables                  0    |    |    |    |    |    |    |    |    |    |    |
storage areas           0    |    |    |    |    |    |    |    |    |    |    |
database rootfile       0    |    |    |    |    |    |    |    |    |    |    |
recovery journals       0    |    |    |    |    |    |    |    |    |    |    |
transactions            1    *    |    |    |    |    |    |    |    |    |    |
hot standby             0    |    |    |    |    |    |    |    |    |    |    |
database                0    |    |    |    |    |    |    |    |    |    |    |
                             |    |    |    |    |    |    |    |    |    |    |
                             |    |    |    |    |    |    |    |    |    |    |
                             |    |    |    |    |    |    |    |    |    |    |
                             |    |    |    |    |    |    |    |    |    |    |
```

```
                             |    |    |    |    |    |    |    |    |    |    |
                             |    |    |    |    |    |    |    |    |    |    |
-----------------------------+----+----+----+----+----+----+----+----+----+----+
Exit Help Menu Set_rate Write !
```

6.1.1 Stall Messages Include Transaction State

RMU Show Statistics Stall Messages screens display the process current transaction state (R for read-only or W for read / write) along with the stall information.

6.1.2 Stall Messages Screen Allows Wildcards

The RMU Show Statistics command Stall Messages screen Filter on-screen menu option allows the use of wildcards in the filtering criteria.

The pattern string can contain either one or both of the two wildcard characters, asterisk (*) and percent (%). The asterisk character is mapped to zero or more characters. The percent character is mapped to only one character.

6.1.3 Stall Message Log Provides Real-Time Lock Information

The RMU Show Statistics Stall Message Logging facility now provides real-time lock information when the displayed stall is on a lock or locked object.

The following example shows the new output of a sample stall messages log file.

```
 Oracle Rdb V7.1 Performance Monitor Stall Log
 Database USR1$:[WORK.STATS]MF_PERSONNEL.RDB;1
 Stall Log created 30-SEP-1997 07:01:15.64
2AA8587A:1 08:11:54.27 reading pages 11:7534416 to 3:78
2AAA9E7B:1 08:11:54.31 waiting for async-write of pages 5:1412 to 5:1412
2AA810A7:1 08:11:54.29 waiting for page 5:876 (PW)
           State... Process.ID Process.name... Lock.ID. Rq Gr Queue page 876
           Blocker: 2AAA9E7B   RICK10......... 7D00562C PR PR Grant
           Waiting: 2AA810A7   RICK13......... 71002E7D PW NL Wait
2AA8587A:1 08:11:55.34 waiting for page 5:1303 (PW)
           State... Process.ID Process.name... Lock.ID. Rq Gr Queue page 1303
           Owner:   2AA7D07C   RICK11......... 31007E07 PR CR Grant
           Blocker: 2AAA9E7B   RICK10......... 5A00BA0E PR PR Grant
           Waiting: 2AA8587A   RICK9.......... 5C005FAD PW CR Cnvrt
2AAA9E7B:1 08:11:55.37 locking page 5:565
2AA810A7:1 08:11:55.38 reading pages 5:912 to 5:914
2AAA9E7B:1 08:11:57.77 waiting for page 5:1303 (PW)
           State... Process.ID Process.name... Lock.ID. Rq Gr Queue page 1303
```

```
           Owner:   2AA810A7   RICK13......... 0C007752 PR CR Grant
           Blocker: 2AA8587A   RICK9.......... 2D001C3D PR PR Grant
           Waiting: 2AAA9E7B   RICK10......... 47003DC3 PW CR Cnvrt
2AA7D07C:1 08:11:57.78 reading pages 5:1337 to 5:1339
2AA8587A:1 08:11:57.86 reading pages 5:330 to 5:332
2AA7D07C:1 08:11:57.86 waiting for page 5:1413 (PR)
           State... Process.ID Process.name... Lock.ID. Rq Gr Queue page 1413
           Blocker: 2AAA9E7B   RICK10......... 6A002CBB PW PW Grant
           Owner:   2AA8587A   RICK9.......... 6F008623 PR CR Grant
           Waiting: 2AA7D07C   RICK11......... 1F007B4D PR NL Wait
.
.
.
```

6.1.4 Stall Messages Log Displays Stall Duration Information

The RMU Show Statistics command Stall Messages Logging facility has been enhanced to provide the information necessary to determine stall duration.

First, the current time has been added to each stall message. This allows you to determine the stall duration at that point, because the stall start time is also displayed.

Second, a new RMU Show Statistics qualifier has been added: Option=Verbose. This qualifier causes the stall message logging facility to report a stall message at each interval, even if the message has been previously reported.

Note: Use of the Option=Verbose qualifier can result in an enormous stall messages log file. Ensure that adequate disk space exists for the log file when you use this qualifier.

You can enable or disable the stall messages logging verbose option at run time by using the Tools menu and pressing the exclamation point (!) key.

You can also specify the verbose option in the configuration file, using the STALL_LOG_VERBOSE variable. (See Section 6.6 for information on configuration files.) Valid keywords are ENABLED or DISABLED.

The following example shows a stall messages log file, in verbose mode, for a database where four processes are all stalled on the same lock. Note that the first stall message already indicates a 25-minute stall.

```
 Oracle Rdb V7.1 Performance Monitor Stall Log
```

```
 Database USR1$:[WORK.STATS]MF_PERSONNEL.RDB;1
 Stall Log created  2-OCT-1997 09:26:15.19
09:26:15.19 2AA8C6D7:1 09:01:01.29 waiting for logical area 58 (CW)
            State... Process.ID Process.name... Lock.ID. Rq Gr Queue logical
area 58
            Blocker: 2AA00443   RICK2.......... 7300845F PW PW Grant
            Waiting: 2AA8C6D7   RICK6.......... 4E008184 CW NL Cnvrt
            Waiting: 2AA912D8   RICK7.......... 5D0034F2 CW NL Cnvrt
            Waiting: 2AA3BADC   RICK8.......... 0700115F CW NL Cnvrt
            Waiting: 2AA43ADE   RICK9.......... 4700AE41 CW NL Cnvrt
09:26:15.19 2AA3BADC:1 09:01:01.37 waiting for logical area 58 (CW)
            State... Process.ID Process.name... Lock.ID. Rq Gr Queue logical
area 58
            Blocker: 2AA00443   RICK2.......... 7300845F PW PW Grant
            Waiting: 2AA8C6D7   RICK6.......... 4E008184 CW NL Cnvrt
            Waiting: 2AA912D8   RICK7.......... 5D0034F2 CW NL Cnvrt
            Waiting: 2AA3BADC   RICK8.......... 0700115F CW NL Cnvrt
            Waiting: 2AA43ADE   RICK9.......... 4700AE41 CW NL Cnvrt
09:26:15.19 2AA912D8:1 09:01:01.32 waiting for logical area 58 (CW)
            State... Process.ID Process.name... Lock.ID. Rq Gr Queue logical
area 58
            Blocker: 2AA00443   RICK2.......... 7300845F PW PW Grant
            Waiting: 2AA8C6D7   RICK6.......... 4E008184 CW NL Cnvrt
            Waiting: 2AA912D8   RICK7.......... 5D0034F2 CW NL Cnvrt
            Waiting: 2AA3BADC   RICK8.......... 0700115F CW NL Cnvrt
            Waiting: 2AA43ADE   RICK9.......... 4700AE41 CW NL Cnvrt
.
.
.
```

The lock information is only displayed once per stall, even in verbose mode, to minimize the output file size.

6.2 Logical Area Screens

The Logical_Area qualifier has been added to the RMU Show Statistics command to allow you to display Logical Area statistics. This qualifier is set by default. A logical area is a table, B-tree index, or hash index. This enhancement provides the means to drill down to the statistical information for a specific table or index.

The following screen is an example of the EMPLOYEES table in the MF_PERSONNEL database:

```
Node: ALPHA3         Oracle Rdb V7.1   Performance Monitor 15-NOV-1996 07:28:18
Rate: 1.00 Second           Logical Area Statistics         Elapsed: 00:55:54.35
```

```
Page: 1 of 1    USER1:[SPANDERSON.WORK.STATS]MF_PERSONNEL.RDB;1     Mode: Online
--------------------------------------------------------------------------------
                         Table EMPLOYEES in EMPIDS_OVER
statistic.........      rate.per.second............ total....... average......
name..............      max..... cur..... avg....... count....... per.trans....
record marked                 14        0        0.0          18           0.0
record fetched                14        0        0.7        2645           0.9
    fragmented                 0        0        0.0           0           0.0
  record stored                9        0        0.7        2621           0.9
     fragmented                0        0        0.0           0           0.0
  pages checked                9        0        0.7        2621           0.9
       saved IO                2        0        0.0         162           0.0
      discarded                0        0        0.0           0           0.0
record erased                  5        0        0.0           6           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
```

In the preceding screen, the statistical information displayed pertains specifically to the EMPLOYEES table within the EMPIDS_OVER storage area. The following screen is an example of the EMPLOYEES_HASH hash index in the MF_PERSONNEL database:

```
Node: ALPHA3         Oracle Rdb V7.1    Performance Monitor 15-NOV-1996 07:47:11
Rate: 1.00 Second           Logical Area Statistics         Elapsed: 01:14:47.77
Page: 1 of 1    USER1:[SPANDERSON.WORK.STATS]MF_PERSONNEL.RDB;1     Mode: Online
--------------------------------------------------------------------------------
                       Hash EMPLOYEES_HASH in EMPIDS_OVER
statistic.........      rate.per.second............ total....... average......
name..............      max..... cur..... avg....... count....... per.trans....
hash insertions                0        0        0.0           0           0.0
     duplicates                0        0        0.0           0           0.0
hash deletions                 0        0        0.0           0           0.0
    duplicates                 0        0        0.0           0           0.0
hash scans                     0        0        0.0           0           0.0
hash index fetches             0        0        0.0           0           0.0
  bucket fragments             0        0        0.0           0           0.0
   duplicate nodes             0        0        0.0           0           0.0
--------------------------------------------------------------------------------
Exit Graph Help Menu Options Pause Reset Set_rate Time_plot Unreset Write X_plot
```

The following screen is an example of the RDB$NDX_REL_NAME_N B-tree index in the MF_PERSONNEL database:

```
Node: ALPHA3         Oracle Rdb V7.1   Performance Monitor 15-NOV-1996 07:49:21
```

```
Rate: 1.00 Second           Logical Area Statistics        Elapsed: 01:16:57.55
Page: 1 of 1    USER1:[SPANDERSON.WORK.STATS]MF_PERSONNEL.RDB;1     Mode: Online
--------------------------------------------------------------------------------
                     Btree RDB$NDX_REL_NAME_N in RDB$SYSTEM
statistic.........      rate.per.second............ total....... average......
name..............      max..... cur..... avg....... count....... per.trans....
node fetches                  1        0        1.5         7135           2.4
 leaf fetches                 0        0        0.6         3039           1.0
 dup. fetches                 0        0        0.2         1189           0.4
index lookups                 0        0        0.1          792           0.2
index scans                   0        0        0.4         2115           0.7
 primary entries              0        0        0.0            0           0.0
 dup. entries                 0        0        0.0            0           0.0
node insertions               0        0        0.0            0           0.0
 root insertions              0        0        0.0            0           0.0
 leaf insertions              0        0        0.0            0           0.0
 dup. insertions              0        0        0.0            0           0.0
node creations                0        0        0.0            0           0.0
 root splits                  0        0        0.0            0           0.0
 leaf creations               0        0        0.0            0           0.0
 dup. creations               0        0        0.0            0           0.0
index creations               0        0        0.0            0           0.0
node removals                 0        0        0.0            0           0.0
 root removals                0        0        0.0            0           0.0
 leaf removals                0        0        0.0            0           0.0
 dup. removals                0        0        0.0            0           0.0
node deletions                0        0        0.0            0           0.0
 leaf deletions               0        0        0.0            0           0.0
 dup. deletions               0        0        0.0            0           0.0
index destructions            0        0        0.0            0           0.0
--------------------------------------------------------------------------------
Exit Graph Help Menu Options Pause Reset Set_rate Time_plot Unreset Write X_plot
```

Note that the type of information displayed depends on the type of the logical area. Also, the graph, time plot, and scatter plot options are available for all fields in the display. The Logical Area screens are selected using the Logical Area Information option of the main menu. Currently, there is only one logical area screen, the Logical Area Statistics screen.

After you select the logical area screen, a menu of available logical areas is displayed. This menu is sorted in alphabetic order. If the RMU Show Statistics utility has information about the type of a particular logical area, then the type will be displayed in the menu option. The type does not affect the sort order. Note that all logical areas, including the system relations and indexes, are displayed in the menu.

After you select a logical area, the RMU Show Statistics utility identifies the type of the selected logical area. If the type cannot be determined, you are prompted to enter the logical area type. The following restrictions and warnings apply:

- The Logical Area statistics are not written to the binary output file. In addition, the Logical Area statistics screens are not available during binary input file replay.
- Currently, there is no mechanism to display aggregate information for a partitioned logical area. All statistical information displayed is on a per-storage-area basis.
- A summary information screen of all logical areas is not currently available.
- The number of lines of statistical information displayed depends on the size of the terminal or window. In particular, the B-tree index displays up to 24 lines of statistical information. On a standard 24x80 character terminal, this means that only the first 15 lines of statistics are displayed.

Note: Any table or index created prior to Rdb7 ECO 1 (V7.0.0.1) requires that you enter the appropriate logical area type. If you specify an incorrect logical area type (for example, you specify a table as being a hash index), then inappropriate statistics fields are displayed.

6.2.1 Reducing the Amount of Memory Consumed by Display

The RMU Statistics command consumes approximately 13,000 bytes of virtual memory per logical area. (The number of logical areas is determined by the largest logical area identifier, not the actual number of areas.)

This can result in the RMU Show Statistics command consuming large amounts of virtual memory, even if you do not want to review logical area statistics information.

Use the Nological_Areas qualifier to the RMU Show Statistics command to indicate that you do not want to display logical area statistics information. When you specify the Nological_Areas qualifier, the virtual memory for logical area statistics information presentation is not acquired.

When you specify the Nological_Areas qualifier do not also specify the Nolog qualifier. If you specify both qualifiers, the logical area statistics information continues to be collected.

There is no corresponding configuration variable. The Nological_Areas qualifier cannot be modified at run time.

The Logical_Areas qualifier is the default.

6.2.2 Logical Area Menu Filter Option

Previously, using the RMU Show Statistics utility Logical Area menu could be difficult if a production database contained hundreds or thousands of logical areas. Now, you can control the contents of the Logical Area menu by using wildcard selection criteria.

The Logical Area menu Filter option that you access from the Tools menu (by entering the exclamation point (!) character) lets you specify a search pattern containing wildcards.

Note: The specified pattern must match at least one logical area, or the pattern is rejected.

The filtered logical area menu is only available when you display all logical areas. It is not available if you select the Display Application Logical Areas option from the Tools menu.

The specified pattern string can contain either one or both of the two wildcard characters: the asterisk (*) and the percent sign (%). The asterisk is mapped to zero or more characters. The percent sign is mapped to only one character. For example, the pattern "*EMP*" finds any logical area containing the text "EMP". The pattern "EMP*" finds only those logical areas whose name starts with "EMP".

6.3 Stall Messages

In previous releases of Oracle Rdb, stall messages (the messages that are displayed on the Stall Messages and Active User Stall Messages screens) were written and formatted by each process attached to the database. Then the RMU Show Statistics process displayed the messages on the screen. This behavior has been changed. Starting with Oracle Rdb release 7.0.1 ECO 1, the RMU Show Statistics process creates and formats the stall message strings. Individual processes attached to the database quickly record information about the stall, but perform no message formatting.

Moving the work of formatting the stall messages to the RMU Show Statistics process reduces CPU usage for all processes attached to the database. Less CPU resources are consumed because stall message information is never formatted by the processes attached to the database (whether or not RMU Show Statistics is being run).

However, a process running the RMU Show Statistics command may use more CPU resources than in previous releases when it is formatting and displaying the Stall Messages and Active User Stall Messages screens. But, moving the formatting work to the process running RMU Show Statistics from the processes attached to the database should result in an overall reduction in the CPU resources used.

6.4 Lock Timeout and Lock Deadlock Logging History Screens

Tools like the RMU Show Statistics utility can help solve a current problem within the database. But most problems are solved within a limited time frame. If you run Oracle Rdb in a 7x24 hour environment, it is not always possible to have an expert readily available. This means you cannot trace a problem until after it has occurred and been solved.

This is especially true when you analyze lock timeouts and deadlocks. Once the event has occurred, all information regarding the lock event is gone. Frequently, a lock deadlock of interest is superseded by another lock deadlock.

The RMU Show Statistics utility provides the Lock Timeout History and Lock Deadlock History screens. However, these screens only record the last timeout or deadlock for the processes. There is no method to record each lock timeout or deadlock as it occurs.

The RMU Show Statistics utility has been enhanced to provide two new command qualifiers: Lock_Timeout_Log=file-spec and Deadlock_Log=file-spec.

The file-spec is the name of the file to which all lock timeout and lock deadlock messages are logged. The lock timeout and lock deadlock messages are written in human-readable format similar to the Lock Timeout History and Lock Deadlock History screens. The header region of the lock timeout and lock deadlock log contains three lines. The first line indicates that the RMU Show Statistics utility created the log file. The second line identifies the database. The third line identifies the date and time the log was created.

The main body of the stall log contains three columns:

- The first column contains the process ID and stream ID that experienced the lock timeout or deadlock.
- The second column contains the time when the timeout or deadlock occurred; however, the date is not displayed.
- The third column contains the timeout or deadlock message describing the affected resource; this message is similar to the originating stall message.

For example:

```
2EA00B52:34 14:25:46.14 - waiting for page 5:751 (PR)
```

If any lock timeouts or lock deadlocks are missed for a particular process (usually because the recording interval is too large), then the number of missed lock timeouts or deadlocks is displayed in brackets after the message. For example:

```
2EA00B52:34 14:25:46.14 - waiting for page 5:751 (PR) [1 missed]
```

Only one message per occurring lock timeout or deadlock is logged.

The lock timeout or deadlock messages are written at the specified screen refresh rate, determined by specifying the Time qualifier, or online using the Set_rate on-screen menu option. Using a larger refresh rate minimizes the size of the file, but results in a large number of missed lock timeout or deadlock messages. Using a smaller refresh rate produces a large log file, but contains a much finer granularity of lock timeout or deadlock messages.

The affected LockID is not displayed, because this is meaningless information after the lock deadlock has completed.

Note that you do not need to be displaying the Lock Timeout History or Lock Deadlock History screens to record the stall messages to the stall log. These logs are maintained regardless of which screen, if any, is displayed.

Note that the Lock_Timeout_Log and Deadlock_Log qualifiers are separate and distinct from each other. They can be used together or separately, as desired.

Using the Lock_Timeout or Deadlock qualifiers, you can construct a Lock Event Logging server. The following OpenVMS DCL script shows how to create a server that logs both lock timeout and lock deadlock events on the mf_personnel database for the next 15 minutes:

```
$ RMU/SHOW STATISTICS /NOHISTOGRAM /TIME=1 /NOINTERACTIVE -
_$ /LOCK_TIMEOUT_LOG=TIMEOUT.LOG /DEADLOCK_LOG=DEADLOCK.LOG -
_$ /NOBROADCAST /UNTIL="+15:00" MF_PERSONNEL
```

Note: Using the Time=1 or Time=-50 qualifier appears to produce a reasonable log while minimizing the impact on the system.

Use the Tools menu (displayed when you press the exclamation point (!) key from any screen) to enable or disable the lock timeout and lock deadlock logging facility while the RMU Show Statistics utility is running. However, note that the lock timeout log and lock deadlock log are not available during binary file replay.

The following sample Lock Deadlock log was produced using a refresh rate of 1.00 seconds:

```
       Oracle Rdb V7.1 Performance Monitor Lock Deadlock Log
       Database USER1:[SPANDERSON.WORK.STATS]MF_PERSONNEL.RDB;1
                              Lock Deadlock Log created  7-FEB-1997 14:20:40.57
2EA00B52:33 14:20:46.22 - waiting for page 5:924 (PW)
2EA04F3A:34 14:20:45.09 - waiting for page 5:792 (PR)
2EA00B52:33 14:20:46.98 - waiting for page 5:1228 (PW)
2EA04F3A:34 14:20:48.19 - waiting for page 5:1086 (PW)
2EA00B52:33 14:20:48.20 - waiting for page 5:1244 (PR)
2EA00B52:33 14:20:58.34 - waiting for page 5:638 (PR)
2EA04F3A:34 14:20:59.17 - waiting for page 5:638 (PW)
2EA04F3A:34 14:21:00.47 - waiting for page 5:482 (PR)
2EA00B52:33 14:21:02.23 - waiting for page 5:661 (PR)
2EA00B52:33 14:21:08.56 - waiting for page 5:523 (PW)
2EA00B52:33 14:21:09.33 - waiting for page 5:918 (PW)
2EA00B52:33 14:21:10.05 - waiting for page 1:976 (PW)
2EA04F3A:34 14:21:15.91 - waiting for page 5:623 (PW)
2EA04F3A:34 14:21:18.63 - waiting for page 5:794 (PW)
2EA00B52:34 14:25:50.45 - waiting for page 5:1086 (PW)
                  .
                  .
                  .
```

6.5 User-Defined Events

Through a configuration file, you can specify special events and what action is to be performed when an event occurs. An event is signaled when a statistic exceeds a user-defined set of thresholds. The EVENT variable value is a free-format description of the user-specified event. The format is defined as follows:

```
EVENT="operation statistic_name event_name [attribute_list]"
```

operation

Identifies what action is to be performed. The operation keywords are the following:

- ENABLE - enables a new event or changes an existing event definition
- DISABLE - disables an existing event, typically used when importing a new configuration file

statistic_name

Identifies the particular valid statistic field for which the event is to be enabled or disabled. Note that some statistic names are valid only when certain database attributes are enabled, such as global buffers or record caching. You can find the name of a particular statistic field on the screen of interest. When statistic names contain multiple words, such as process attaches, enclose the statistic name in either single quotation marks or double quotation marks; failure to use quotation marks with the statistic name may result in a parsing error.

event_name

Identifies the particular event threshold for the specified statistic to be enabled or disabled. Up to eight different thresholds can be specified for a particular statistic field. The event_name keywords are the following:

- MAX_CUR_TOTAL - the maximum total value collected so far
- MIN_CUR_RATE - the lowest rate currently being sustained
- MAX_CUR_RATE - the highest rate currently being sustained
- MAX_RATE - the maximum current rate collected so far
- MIN_AVG_RATE - the lowest average rate
- MAX_AVG_RATE - the highest average rate
- MIN_PER_TX - the lowest per-transaction rate
- MAX_PER_TX - the highest per-transaction rate

attribute_list

The optional attribute_list provides information about enabled events; these attributes are ignored when you disable an event. The attribute_list keywords are the following:

- INITIAL value - Defines the initial value of the event threshold. The default value is 0 for MAX_XXX events and over a billion for MIN_XXX events. The default value guarantees that at least one event will be signaled, thereby initializing the new current threshold value.
- EVERY value - Defines the value by which the initial threshold will be incremented or decremented when an event is signaled. If this value is 0 (the default value), then the event is signaled just once.
- LIMIT value - Defines the maximum number of times that the event can be signaled. If the value is 0 and the EVERY keyword is specified with a nonzero value, then events can be signaled indefinitely. The default value of the LIMIT value is 0.
- SKIP value - Defines the number of events to skip before invoking the program.
- NOTIFY oper_class_list - Defines the quoted, comma-separated list of operators that are to be notified for all events defined on the specified statistic.
- INVOKE program - Defines the quoted, user-supplied program that is to be invoked for all events defined on the specified statistic.
- AREA area-name - Allows you to specify the name of a storage area. When this keyword is specified, the statistics field selected must be from the IO Statistics (by file) or Locking Statistics (by file) screens.

  The AREA keyword is available when you use the RMU Show Statistics command with the Nointeractive qualifier, or when you set the INTERACTIVE configuration variable to FALSE. See Table 6–1 for information on the semantics of this keyword.
- LAREA larea-name - Allows you to specify the name of a logical area, which can be a table, B-tree index, hash index, or BLOB. When this keyword is specified, the statistics field selected must be from the Logical Area screens.

  If the logical area is partitioned across multiple storage areas, the AREA keyword can be used to identify a specific partition to define the event against.

  The LAREA keyword is available when you use the RMU Show Statistics command with the Nointeractive qualifier or when you set the INTERACTIVE configuration variable to FALSE. See Table 6–1 for information on the semantics of this keyword.

Event Semantics

- For an event to be active, you must specify either one or both of the NOTIFY or INVOKE keywords. When you use the INVOKE keyword, the program must be a symbol pointing to the DCL script or image to be invoked. Also note that the INVOKE and NOTIFY definitions apply to all events defined for the specified statistic field. Therefore, specifying either or both of these attributes for multiple event descriptions for the same statistics field means that only the last definition will be used.

- The INVOKE program and NOTIFY operator classes apply to all events defined for the statistic field. Therefore, you need to define these keywords only once per statistic field, no matter how many events you define. Even if you specify multiple programs or operator classes, only the last specified attribute is used.
- Once an event has been signaled, it will be resignaled only if the EVERY keyword was specified with a nonzero value. The current threshold value, originally initialized to the INITIAL value, is incremented for MAX_XXX thresholds and decremented for MIN_XXX thresholds. The MIN_XXX thresholds disable themselves once the INITIAL value reaches 0, while the MAX_XXX thresholds never disable themselves.
- Once an event has been disabled, you can reenable it only by importing a new configuration file or by manually using the Statistics Event Information screen "Re-enable all disabled events configuration" submenu option. The user-defined events are analyzed at the specified screen refresh rate. Multiple events defined for the same statistic field may cause the specified program to be invoked multiple times, once for each affected event.
- If a program is specified to be invoked when an event occurs, the program will be invoked with seven parameters. Some of the parameters contain multiple words; be sure to quote them if the parameters are passed to other utilities. The parameters are the following, where P1 represents the first parameter, P2 the second parameter, and so on:
  - P1: The date and time when the event occurred. (This parameter contains embedded blank spaces.)
  - P2: The statistic field name.
  - P3: The event name.
  - P4: The current event value expressed to the nearest tenth.
  - P5: Either the word "above" or "below."
  - P6: The current event threshold value.
  - P7: The event occurrence count.

  – P8: The optional physical area name and / or logical area name for the statistic field.

  Example 1 shows a configuration file that contains four user-defined event entries.

- Table 6–1 explains the semantics of specifying the AREA and LAREA keywords.

Table 6–1 Semantics of AREA and LAREA Event Attributes

AREA Specified	LAREA Specified	Result
No	No	Regular statistic field used
Yes	No	Storage Area statistic field used
No	Yes	Logical Area statistic field used - all partitions
Yes	Yes	Logical Area statistic field used - single partition

Examples

Example 1 Simple Event Configuration File

```
# This file was generated by the SHOW STATISTIC utility.
DATABASE = "USER1:[SPANDERSON.WORK.STATS]MF_PERSONNEL.RDB;1"
# Generated on  8-APR-1997 15:08:33.72
EVENT_DESCRIPTION="ENABLE transactions MAX_RATE INITIAL 5 EVERY 2 INVOKE'LOG_EVENT'";
EVENT_DESCRIPTION="DISABLE transactions MAX_CUR_RATE";
EVENT_DESCRIPTION="ENABLE transactions MIN_CUR_RATE INITIAL 5 EVERY 2 LIMIT 5";
EVENT_DESCRIPTION="ENABLE 'process failures' MAX_CUR_TOTAL INITIAL 1 EVERY 1 NOTIFY OPER12";
```

In Example 1, an event is enabled for the transactions statistic, specifying a maximum rate threshold starting at 5 and incrementing by 2 whenever the event is signaled. When the event is signaled, the user-supplied program LOG_EVENT will be invoked.

In Example 1, the program LOG_EVENT identifies an OpenVMS symbol that defines the program to be executed. The symbol is defined on OpenVMS as follows:

```
$ LOG_EVENT :== @SYS$SYSTEM:LOG_EVENT.COM
```

If the program fails for any reason, the event will be disabled automatically. The following is an example of the LOG_EVENT.COM DCL script:

```
$ SET NOON
$ OPEN/APPEND/SHARE=READ EVENT_LOG SYS$DISK:[]EVENT.LOG
```

```
$ WRITE EVENT_LOG "''p1' ''p2' ''p3' ''p4' ''p5' ''p6' (count is ''p7')"
$ CLOSE EVENT_LOG
$ EXIT
```

In this example, the DCL script appends the event message to the disk file EVENT.LOG. Note that in the preceding event description example, the second event to enable the MIN_CUR_RATE threshold for the transactions statistic specified neither an INVOKE nor NOTIFY attribute. Because one of these attributes was already specified for the statistic event definition of the preceding transaction, you do not need to specify it again.

When the event is signaled, the designated program or operator class or classes are invoked with a description of the event. For example:

```
22-MAY-1997 13:41:35.17 transactions MIN_PER_TX 1.0 below 4294967295.0 (count is 1)
22-MAY-1997 13:42:06.73 transactions MAX_RATE 14.0 above 1.0 (count is 1)
22-MAY-1997 13:42:07.66 transactions MAX_CUR_TOTAL 17.0 above 2.0 (count is 1)
22-MAY-1997 13:42:08.61 transactions MAX_CUR_RATE 14.0 above 2.0 (count is 1)
22-MAY-1997 13:42:10.58 transactions MAX_RATE 15.0 above 14.0 (count is 2)
22-MAY-1997 13:42:11.45 transactions MAX_CUR_TOTAL 77.0 above 17.0 (count is 2)
22-MAY-1997 13:42:12.39 transactions MAX_CUR_RATE 15.0 above 14.0 (count is 2)
22-MAY-1997 13:42:14.34 transactions MAX_CUR_TOTAL 137.0 above 77.0 (count is 3)
22-MAY-1997 13:42:16.25 transactions MAX_CUR_TOTAL 169.0 above 137.0 (count is 4)
22-MAY-1997 13:42:18.12 transactions MAX_RATE 17.0 above 16.0 (count is 3)
22-MAY-1997 13:42:18.98 transactions MAX_CUR_TOTAL 201.0 above 169.0 (count is 5)
22-MAY-1997 13:42:19.88 transactions MAX_CUR_RATE 17.0 above 16.0 (count is 3)
22-MAY-1997 13:42:21.78 transactions MAX_CUR_TOTAL 260.0 above 201.0 (count is 6)
22-MAY-1997 13:42:23.65 transactions MAX_CUR_TOTAL 294.0 above 260.0 (count is 7)
22-MAY-1997 13:42:25.58 transactions MAX_CUR_TOTAL 326.0 above 294.0 (count is 8)
22-MAY-1997 13:42:27.43 transactions MAX_CUR_TOTAL 356.0 above 326.0 (count is 9)
22-MAY-1997 13:42:28.43 transactions MIN_AVG_RATE 0.1 below 4294967295.0 (count is 1)
22-MAY-1997 13:42:30.52 transactions MAX_CUR_TOTAL 407.0 above 356.0 (count is 10)
22-MAY-1997 13:42:32.36 transactions MAX_CUR_TOTAL 441.0 above 407.0 (count is 11)
22-MAY-1997 13:42:34.29 transactions MAX_CUR_TOTAL 474.0 above 441.0 (count is 12)
22-MAY-1997 13:42:36.23 transactions MAX_CUR_TOTAL 507.0 above 474.0 (count is 13)
22-MAY-1997 13:42:38.12 transactions MAX_CUR_TOTAL 542.0 above 507.0 (count is 14)
22-MAY-1997 13:42:40.13 transactions MAX_CUR_TOTAL 574.0 above 542.0 (count is 15)
```

The current run-time state of the user-defined events can be examined using the new Statistics Event Information screen, located in the Database Parameters submenu. Note that you do not have to be in the screen to signal events.

Consider the following Statistics Event Information screen example:

```
Node: ALPHA3 (1/1/2)    Oracle Rdb V7.1    Perf. Monitor 22-MAY-1997 13:54:44.37
Rate: 1.00 Second         Statistics Event Information      Elapsed: 01:10:46.17
```

```
Page: 1 of 8    USER1:[SPANDERSON.WORK.STATS]MF_PERSONNEL.RDB;1     Mode: Online
--------------------------------------------------------------------------------
Statistic......... Event........ State...     Threshold Every       Current  Cnt
transactions       MAX_RATE      enabled            1.0     2           1.0    0
transactions       MAX_CUR_TOTAL enabled         8194.0     2        8194.0    1
transactions       MIN_CUR_RATE  disabled           1.0     2           0.0    1
transactions       MAX_CUR_RATE  enabled            2.0     2           0.0    0
transactions       MIN_AVG_RATE  disabled           1.9     0           1.9    1
transactions       MAX_AVG_RATE  enabled            3.0     2           1.9    0
transactions       MIN_PER_TX    disabled           1.0     2           1.0    1
transactions       MAX_PER_TX    enabled            4.0     2           1.0    0
--------------------------------------------------------------------------------
Config Exit Help Menu >next_page <prev_page Options Set_rate Write !
```

Note that the screen contains the Full on-screen menu option. Using this option displays two lines of information per event, including the INVOKE program file specification, if any.

Example 2 Defining Events on Database Objects

The following code demonstrates how to define an event on a storage area statistic:

```
EVENT_DESCRIPTION="ENABLE '  (Asynch. reads)' MAX_CUR_TOTAL \
AREA EMPIDS_OVER \
INITIAL 6 EVERY 10 LIMIT 100 INVOKE DB_ALERT";
```

The following code demonstrates how to define an event on a table. Note that this event is defined across all partitions of the table.

```
EVENT_DESCRIPTION="ENABLE 'pages checked' MAX_CUR_TOTAL \
AREA EMPLOYEES \
INITIAL 1 EVERY 1 LIMIT 100 INVOKE DB_ALERT";
```

The following code demonstrates how to define an event on a single-partition of a partitioned table:

```
EVENT_DESCRIPTION="ENABLE 'pages checked' MAX_CUR_TOTAL \
LAREA EMPLOYEES AREA EMPIDS_LOW \
INITIAL 3 EVERY 7 LIMIT 100 INVOKE DB_ALERT";
```

The Statistics Event Information screen has been enhanced to identify the physical area ID and logical area ID for each event. The area identifiers are displayed when you use Full display mode. For example, if you use the code in Example 2, the screen would appear as follows:

```
Node: ALPHA3 (1/1/2)    Oracle Rdb V7.1    Perf. Monitor 21-OCT-1997 13:41:50.06
```

```
Rate: 1.00 Second        Statistics Event Information      Elapsed: 02:30:21.57
Page: 1 of 1    USER:[SPANDERSON.WORK.STATS]MF_PERSONNEL.RDB;1     Mode: Online
--------------------------------------------------------------------------------
Statistic......... Event........ State...     Threshold Every       Current  Cnt
Program/Operator.Notification............................ Parea Larea Rem Limit
synch data reads   MAX_CUR_TOTAL enabled          228.0    11         228.0    1
DB_ALERT (@SYS$DISK:[]EVENT.COM)                              0     0   0   100
locks requested    MAX_CUR_TOTAL enabled          406.0    10         406.0    1
DB_ALERT (@SYS$DISK:[]EVENT.COM)                              5     0   0   100
pages checked      MAX_CUR_TOTAL enabled            3.0     7           0.0    0
DB_ALERT (@SYS$DISK:[]EVENT.COM)                              3    56   0   100
pages checked      MAX_CUR_TOTAL enabled            4.0     8           0.0    0
DB_ALERT (@SYS$DISK:[]EVENT.COM)                              4    57   0   100
pages checked      MAX_CUR_TOTAL enabled        10717.0     9       10717.0    1
DB_ALERT (@SYS$DISK:[]EVENT.COM)                              5    58   0   100
pages checked      MAX_CUR_TOTAL enabled        10717.0     1       10717.0    1
DB_ALERT (@SYS$DISK:[]EVENT.COM)                              0     2   0   100
--------------------------------------------------------------------------------
Brief Config Exit Help Menu >next_page <prev_page Options Pause Set_rate Write
```

If an event on a storage area or logical area is encountered, the storage area name, the logical area name, or both are passed as the eighth parameter to the invocation program. For example, the DB_ALERT program defined in the previous example causes the DCL script EVENT.COM to be executed. This DCL script appends the raised event to a log file; for example:

```
$ SET NOON
$ OPEN/APPEND/SHARE=READ EVENT_LOG SYS$DISK:[]EVENT.LOG
$ WRITE EVENT_LOG "''P1' ''P2' ''P3' ''P4' ''P5' ''P6' COUNT IS ''P7' ''P8'"
$ CLOSE EVENT_LOG
$ EXIT
```

Note that the P8 parameter is either null ("") or contains the name of the target storage area or logical area. The following is an example of the log file output:

```
20-OCT-1997 14:02:21.41 pages checked MAX_CUR_TOTAL 6.0 above 4.0 count is 1
EMPIDS_MID.EMPLOYEES
20-OCT-1997 14:02:22.16 pages checked MAX_CUR_TOTAL 32820.0 above 5.0 count is 1
EMPIDS_OVER.EMPLOYEES
```

Note that when both the storage area and logical area names are specified, they are separated by a period (.) character.

Example 3 Step-by-Step Event Example

Suppose that you want to be sent electronic mail (e-mail) whenever an application process on the database terminates prematurely. You can accomplish this using events, as the following steps describe:

1. Identify the operation.

   This example will enable a new event.

2. Identify the statistic name to which the event will be assigned.

   This example uses the process failures statistic from the Recovery Statistics screen (located in the AIJ Information submenu). This statistic is available even if you are not using after-image journaling.

3. Identify the event name to use.

   This example uses the MAX_CUR_TOTAL event name, because this represents the current number of processes that have failed.

4. Identify the event attributes to use.

   This is probably the hardest part of defining an event. To be alerted to any process failure, set the INITIAL attribute to 0. To be alerted to every process failure, set the EVERY attribute to 1 and the LIMIT attribute to 0.

5. Define how to be alerted about the event.

   To be sent mail, use the INVOKE keyword. Invoking a program on OpenVMS requires that you define a DCL symbol to identify the actual DCL script; for example:

   ```
   $ DBR_LOGGER :== @SYS$DISK:[]DBR_LOGGER.COM
   ```

6. Write the program to be invoked.

   To be sent mail with a description of what event actually occurred, this example uses the following DCL script DBR_LOGGER.COM:

   ```
   $ SET NOON
   $ CREATE /NOLOG SYS$SCRATCH:DBR_LOGGER.TMP
   EOD
   $ OPEN /WRITE DBR_LOGGER SYS$SCRATCH:DBR_LOGGER.TMP
   $ WRITE DBR_LOGGER "''P1' ''P2' ''P3' ''P4' ''P5' ''P6' COUNT IS ''P7'"
   $ CLOSE DBR_LOGGER
   $ MAIL SYS$SCRATCH:DBR_LOGGER.TMP ANDERSON /SUBJECT="DBR NOTIFICATION"
   $ DELETE /NOLOG SYS$SCRATCH:DBR_LOGGER.TMP;*
   $ EXIT
   ```

7. Combine all of this information into the configuration file entry.

```
EVENT_DESCRIPTION="ENABLE 'process failures' MAX_CUR_TOTAL INITIAL 0 EVERY 1 INVOKE DBR_LOGGER";
```

8. Run the RMU Show Statistics command using the configuration file.

```
$ RMU/SHOW STATISTICS /CONFIG=CONFIG.CFG MF_PERSONNEL
```

6.6 Creating and Using a Configuration File

The ability to import and export a configuration file has been added to the RMU Show Statistics command. The configuration file is a file in human-readable format that can be created using any editor, or can be automatically generated from the RMU Show Statistics command using the current run-time configuration settings.

The configuration file is processed by the RMU Show Statistics command prior to opening the database or the binary input file, if the new optional Configure=file-spec qualifier is used. If you do not specify this qualifier, all of the variables are the defaults based on command-line qualifiers and logical names. The default configuration file type is .cfg. The configuration file is processed in two passes. The first pass occurs before the database is opened and processes most of the configuration file entries. The second pass occurs after the database is opened, and processes those variables that are database-dependent, such as the CUSTOMER_LINE_n variable.

6.6.1 Configuration File Syntax

Each entry in the configuration file uses the following general format:

```
variable=value;
```

The equal sign (=) separating the variable and value is required. Each definition is terminated using the semicolon (;).

A comment is specified using either the number sign (#) or exclamation point (!) character, and continues to the end of the current line. A comment can occur anywhere in a line, but always terminates that line.

Blank lines are ignored.

The variable = value is free-format. That is, spacing is not relevant to the parsing of the token. For example, the entry STALL_LOG="STALL.LOG"; could be entered in the configuration file as:

```
STALL_LOG
```

```
=
"STALL.LOG"
```

However, for purposes of readability, this type of formatting is not recommended.

Also, multiple entries can be put on a single line, although this is also not recommended for purposes of readability. For example, the following entry is permitted:

```
STALL_LOG="STALL.LOG"; DBKEY_LOG="DBKEY.LOG";
```

Long configuration file lines can be continued on the next line by terminating the line with a backslash (\). Lines can be continued up to 2048 characters, even within quoted values; for example:

```
EVENT_DESCRIPTION="ENABLE 'pages checked' MAX_CUR_TOTAL \
INITIAL 7 \
EVERY 11 \
LIMIT 100 \
INVOKE DB_ALERT";
```

This is not limited to just the EVENT_DESCRIPTION variable; it can be used for any configuration variable. Also, comments can be embedded in continued lines if they start at the beginning of the next line. For example, consider the following two event descriptions:

```
EVENT_DESCRIPTION="ENABLE '  (Asynch. reads)' MAX_CUR_TOTAL \
! This will work as expected
AREA EMPIDS_OVER \
INITIAL 6 EVERY 10 LIMIT 100 INVOKE DB_ALERT";

EVENT_DESCRIPTION="ENABLE '  (Asynch. reads)' MAX_CUR_TOTAL \
AREA EMPIDS_OVER ! This will NOT work as expected \
INITIAL 6 EVERY 10 LIMIT 100 INVOKE DB_ALERT";
```

In the first event description, the only code included in the comment is the phrase, "This will work as expected". However, in the second event description, the line-continuation character is part of the comment. Therefore, when the last line of code is processed, it is not processed as a continuation of the second event description, but instead as an independent statement. This results in a syntax error.

6.6.2 Creating Configuration Files

The configuration file can be created using the editor of your choice. A new configuration file can also be automatically exported by pressing the exclamation point (!) key to open the Tools menu, and then selecting the Save Current Configuration option.You will be prompted to enter the name of the new configuration file. The default configuration file type is .cfg. For portability, comments are generated using the number sign (#) character.

6.6.3 Importing Configuration Files

A new configuration file can be imported at any time by selecting the Tools menu, pressing the exclamation point (!) key, and then selecting the Import Configuration Settings option. You will be prompted to enter the name of the new configuration file. The default configuration file type is .cfg.

6.6.4 Nested Configuration Files

Configuration files can be nested to any depth, using the special INCLUDE variable. Using nested configuration files allows you to create hierarchies of configuration variable definitions.

Note: When using nested configuration files, remember that the last definition of a variable is the one that is used by the RMU Show Statistics command.

6.6.5 Variable Semantics

The variable is either a predefined configuration parameter known to the RMU Show Statistics command, or a user-defined variable whose value will presumably be used later as the value of some other predefined configuration parameter. The complete list of predefined variables is defined in Section 6.6.12. All variable names must start with an alphabetic character. The variable name is case-insensitive. All variables known to the RMU Show Statistics command have defaults, which are listed in Table 6–2. It is not necessary to explicitly specify every variable. It is generally recommended that you explicitly specify only those variables whose default value is not acceptable to your application. Most configuration file variables replace existing command-line qualifiers or logical names.

6.6.6 Variable Types

Configuration variables have seven types. These types dictate how the corresponding atomic value is used. The variable types are:

- Numeric - This is the most common type of configuration variable. Some, but not all, numeric variables have minimum and maximum values. Also, some, but not all, numeric variables have scale values that dictate the number of fractional digits allowed. For example, a variable with a scale of 2 can be specified in terms of hundredths. A scale of 0 designates that only a whole number is allowed (no decimal positions).
- Boolean - Boolean variables can use the values TRUE and FALSE or ENABLED and DISABLED interchangeably. These values are not quoted. You can also use the numeric values 0 and 1.
- String - String variables specify a quoted value, which is typically a file specification. Null string values are specified using an empty set of quotation marks ("").
- Date - Date variables specify a quoted value in the standard operating system date format.
- Command - Command variables cause an action specified by the quoted value to be performed. The most common command variables are the PRINT, PROMPT, and INQUIRE variables. Command variables are not automatically exported.
- Control - Control variables are most commonly used to document interesting values. They do not actually do anything, but they can be useful as the value of another variable. The most common variable is the DATABASE variable.
- User-defined - User-defined variables are used as the values of other types of variables and are treated as simple strings. User-defined variables are not exported.

6.6.7 Value Semantics

The values for numeric variables can be specified using either whole numbers or floating-point numbers, depending on the respective scale of the particular variable. The scale of a variable indicates the number of digits to the right of the decimal place.

If you specify a value whose fractional portion exceeds the scale, then the value is truncated. For example, if you specify 4.25 for a variable whose scale is 1 (meaning 1 digit to the right of the decimal point) then only 4.2 is used.

When the variable value is quoted using double quotation marks (""), the evaluation of the value is treated as a hierarchy of values. The evaluation hierarchy is: variable name, logical name, atomic value. This hierarchy means that every value is first iteratively treated as another variable, then iteratively treated as a logical name, then finally as an atomic value.

Note that you can also specify a single-word string value without the double quotation marks, if desired. For example, the value EXAMPLE can be specified using either double quotation marks or no quotation marks, because EXAMPLE is a single-word string. However, the value HERE AND NOW must be enclosed with either single or double quotation marks.

Quotation marks can also be nested, if they alternate. Note the pairings of the quotation marks in the following example:

```
QUOTE_TEST = "1 ′2 "3 ′4 "5 ′6 "7 ′8 8′ 7" 6′ 5" 4′ 3" 2′ 1";
             ^  ^  ^  ^  ^  ^  ^  ^---^  ^  ^  ^  ^  ^  ^  ^
             |  |  |  |  |  |  +---------+  |  |  |  |  |  |
             |  |  |  |  |  +---------------+  |  |  |  |  |
             |  |  |  |  +---------------------+  |  |  |  |
             |  |  |  +---------------------------+  |  |  |
             |  |  +---------------------------------+  |  |
             |  +---------------------------------------+  |
             +---------------------------------------------+
```

The hierarchical evaluation semantics allows you to specify the value of one variable as the value of another. In the following example, the user-defined variable TEST is defined with the value 0. The predefined numeric variable CYCLE is defined to the value TEST, which, because it is a variable, will be evaluated as the value 0. Finally, the predefined command variable PRINT will be evaluated as CYCLE, which, because it is also a variable, will be evaluated as the value 0.

```
TEST    = 0
CYCLE   = "TEST"
PRINT   = "CYCLE"
```

Note that, even though the CYCLE variable is a numeric type, you can specify a quoted string value if it ultimately evaluates to a numeric value.

Also note that the hierarchical evaluation of values prevents a quoted value from being the same as the variable name. For instance, in the following configuration file entry, the value INCLUDE cannot be evaluated as the variable name INCLUDE, because this would be recursive:

```
INCLUDE="INCLUDE";
```

Using logical names as the value of variables is useful for certain variables, such as the INCLUDE variable. This allows you to customize your configuration files through the use of logical names while still using a centralized configuration file hierarchy.

Some variables have special keywords for their values. These keywords must be specified in uppercase, exactly as they are described in this manual. Be especially careful when you specify a user-defined variable whose name conflicts with any of the valid keywords. When you use atomic values for keywords, Oracle Corporation recommends that you use single quotation marks to avoid inadvertent evaluation of the keyword as another variable. When the value of a variable is quoted using single quotation marks ('), the value is treated as a simple string.

6.6.8 PRINT and PROMPT Command Variables

The PRINT and PROMPT variables are useful for debugging and tracing execution of the configuration file. For instance, the following example demonstrates how to display the default value of a variable, and then displays the new value initialized from the configuration file:

```
PRINT="CYCLE";
CYCLE=5;
PRINT="CYCLE";
```

The output from the preceding example is the following:

```
# This file was generated by the SHOW STATISTIC utility.
# Generated on 16-APR-1997 10:35:05.48
line 3: variable "PRINT" value "CYCLE = 0"
line 5: variable "PRINT" value "CYCLE = 5"
```

The PRINT variable is also useful for tracing execution of the configuration file. This may be useful when using nested configuration files. For example, the PRINT command can be used to display simple messages to the log file:

```
PRINT="Now initializing CYCLE variable";
CYCLE=5;

The output from the preceding example is the following:
# This file was generated by the SHOW STATISTIC utility.
# Generated on 16-APR-1997 10:35:05.48
line 1: variable "PRINT" value "Now initializing CYCLE variable"
```

Take care when printing out single-word messages, as these might be evaluated as variables. Consider the following example:

```
HERE="there";
```

```
PRINT="here";
PRINT='here';
PRINT="here and there";
```

In this example, the value of the first PRINT variable is there not "here", because "here" is a user-defined variable. The value of the second PRINT variable is "here" because it is a single-quoted simple string. Finally, the value of the third PRINT variable is "here and there" because it is an atomic string.

6.6.9 Variable Value Redirection

Redirecting is an interesting use of a variable, especially a variable whose value is input by the user. Consider the following configuration file example:

```
OVER="AND OUT";
THERE="OVER";
HELLO="THERE";
FOO="HELLO";
```

Setting the variable FOO to the value HELLO causes an implicit redirection of its value to be the final value of all of its own values. This results in the FOO variable being set to the value AND OUT, which is the final value of all values.

The REDIRECT command variable is particularly useful following an INQUIRE command, where you entered the name of another variable to use. By redirecting the value of the variable, you can implement a menu mechanism. Consider the following configuration file example:

```
VALUE1="10";
VALUE2="20";
VALUE3="30";
PROMPT="Enter VALUE1, VALUE2, or VALUE3";
INQUIRE="VALUE";
REDIRECT="VALUE";
PRINT="VALUE";
```

Note how this differs from setting the VALUE variable to itself.

6.6.10 Configuration File Example

The following is an example of a configuration file:

```
# This file was generated by the SHOW STATISTIC utility.
DATABASE = "USER1:[SPANDERSON.WORK.STATS]MF_PERSONNEL.RDB;1"
# Generated on  8-APR-1997 15:08:33.72
```

```
EVENT_DESCRIPTION="ENABLE transactions MAX_RATE INITIAL 5 EVERY 2 INVOKE EVENT";
EVENT_DESCRIPTION="DISABLE transactions MAX_CUR_RATE";
EVENT_DESCRIPTION="ENABLE transactions MIN_CUR_RATE INITIAL 5 EVERY 2 LIMIT 5";
ALARM = 0;
BROADCAST = FALSE;
CYCLE = 0;
INTERACTIVE = TRUE;
HISTOGRAM = FALSE;
REOPEN_INTERVAL = 0;
REFRESH_INTERVAL = 1;
NOTIFY = "OPER1,OPER11";
NOTIFY = "OPER1,OPER11";
STALL_LOG = "stall.log";
TIMEOUT_LOG = "timeout.log";
DEADLOCK_LOG = "deadlock.log";
DBKEY_LOG = "";
TX_DURATION = "TOTAL";
LOGICAL_AREA = "INDIVIDUAL";
STALL_MESSAGE = "ACTUAL";
ACTIVE_USER = "ACTUAL";
CHECKPOINT_TX = "ACTUAL";
CHECKPOINT_SORT = "OLDEST_CHECKPOINT";
AIJ_ARBS_PER_IO = 99.9;
AIJ_BKGRD_ARB_RATIO = 50.5;
AIJ_BLKS_PER_IO = 2.5;
AIJ_SEC_TO_EXTEND = 60.5;
BTR_FETCH_DUP_RATIO = 15.5;
BTR_LEF_FETCH_RATIO = 25.5;
DBR_RATIO = 15.5;
FULL_BACKUP_INTRVL = 6;
GB_IO_SAVED_RATIO = 85.5;
GB_POOL_HIT_RATIO = 85.5;
LB_PAGE_HIT_RATIO = 75.5;
MAX_HASH_QUE_LEN = 2;
MAX_LOCK_STALL = 2.5;
MAX_TX_DURATION = 15.5;
PAGES_CHECKED_RATIO = 10.5;
RECS_FETCHED_RATIO = 20.5;
RECS_STORED_RATIO = 20.5;
RUJ_SYNC_IO_RATIO = 10.5;
VERB_SUCCESS_RATIO = 25.5;
```

6.6.11 Configuration File Log File

If any problem occurs while processing the specified configuration file, the Show Statistics facility automatically creates a log file. The log file name is the same as the configuration file specification with the .cfg file type replaced with a .log file type. For example, if the configuration file specification is config.cfg, then the corresponding log file specification is config.log.

Log file entries have the format: "line #: message". The following example shows a problem trying to include a nested configuration file; the offending command is on line 4 of the configuration file.

```
# This file was generated by the SHOW STATISTIC utility.
# Generated on  6-APR-1997 07:20:00.73
line 4: INCLUDE="config.cfg";
line 4: command "INCLUDE" failed
%COSI-E-FLK, file currently locked by another user
```

You can also specify the Log qualifier, which lists each processed variable to the corresponding log file. Oracle Corporation recommends that you specify this qualifier until you become familiar with using the configuration files.

For example, the following config.log was produced using the Log qualifier in conjunction with the Configure=config.cfg qualifier. Note that, in some cases, the set of valid keywords is included as a comment for your convenience.

```
# This file was generated by the SHOW STATISTIC utility.
DATABASE = "USER1:[SPANDERSON.WORK.STATS]MF_PERSONNEL.RDB;1";
# Generated on 15-APR-1997 15:31:52.92
ALARM = 0;
BROADCAST = TRUE;
CYCLE = 0;
DASHBOARD_UPDATE = TRUE;
INTERACTIVE = TRUE;
HISTOGRAM = FALSE;
REOPEN_INTERVAL = 0;
RESET = FALSE;
REFRESH_INTERVAL = 1;
OUTPUT = "";
SCREEN = "Stall Messages";
STALL_LOG = "stall.log";
TIMEOUT_LOG = "timeout.log";
DEADLOCK_LOG = "deadlock.log";
DBKEY_LOG = "";
CLUSTER_NODES = "";
NOTIFY = "OPER1,OPER11"; # "CENTRAL" "DISKS" "CLUSTER" "SECURITY" "OPER2"
```

```
"OPER3" "OPER4" "OPER5" "OPER6" "OPER7" "OPER8" "OPER9" "OPER10" ...
TX_DURATION = "TOTAL"; # "TOTAL" "READ_WRITE" "READ_ONLY"
LOGICAL_AREA = "INDIVIDUAL"; # "INDIVIDUAL" "AGGREGATE"
STALL_MESSAGE = "ACTUAL"; # "ACTUAL" "ELAPSED"
ACTIVE_USER = "ACTUAL"; # "ACTUAL" "ELAPSED"
CHECKPOINT_TX = "ACTUAL"; # "ACTUAL" "ELAPSED"
CHECKPOINT_SORT = "OLDEST_CHECKPOINT"; # "OLDEST_CHECKPOINT"
"OLDEST_TRANSACTION" "OLDEST_QUIET_POINT"
AIJ_ARBS_PER_IO = 99.9;
AIJ_BKGRD_ARB_RATIO = 50.5;
AIJ_BLKS_PER_IO = 2.5;
AIJ_SEC_TO_EXTEND = 60.5;
BTR_FETCH_DUP_RATIO = 15.5;
BTR_LEF_FETCH_RATIO = 25.5;
DBR_RATIO = 15.5;
FULL_BACKUP_INTRVL = 6;
GB_IO_SAVED_RATIO = 85.5;
GB_POOL_HIT_RATIO = 85.5;
LB_PAGE_HIT_RATIO = 75.5;
MAX_HASH_QUE_LEN = 2;
MAX_LOCK_STALL = 2.5;
MAX_TX_DURATION = 15.5;
PAGES_CHECKED_RATIO = 10.5;
RECS_FETCHED_RATIO = 20.5;
RECS_STORED_RATIO = 20.5;
RUJ_SYNC_IO_RATIO = 10.5;
VERB_SUCCESS_RATIO = 25.5;
```

6.6.12 Predefined Configuration Variable Definitions

This section describes the predefined configuration variables that are known to the RMU Show Statistics command. Table 6–2 lists the predefined variables in alphabetic order. However, note that they are not necessarily be exported in that order.

Table 6–2 Predefined Variable Definitions

Variable Name	Type	Default Value	Min. Value	Max. Value	Scale	Description
ACTIVE_USER	String				1	Specifies the Active User Stall Messages screen configuration options. The valid keywords are ACTUAL and ELAPSED.
AIJ_ARBS_PER_IO	Numeric	2	0		1	Specifies the default online analysis value of AIJ request blocks per AIJ I/O. The RDMS$BIND_STATS_AIJ_ARBS_PER_IO logical name allows you to override the default value of AIJ request blocks per AIJ I/O. The default is 2 blocks. You can also set this threshold from the Configuration submenu in the Performance Monitor Online Analysis facility.
AIJ_BKGRD_ARB_RATIO	Numeric	50	0		1	Specifies the default online analysis value for the background AIJ request block threshold. The RDM$BIND_STATS_AIJ_BKGRD_ARB_RATIO logical name allows you to override the default value for the background AIJ request block threshold. The default value is 50. You can also set this threshold from the Configuration submenu in the Performance Monitor AIJ Analysis screen.
AIJ_BLKS_PER_IO	Numeric	2	0		1	Specifies the default online analysis value of blocks per AIJ I/O. The RDM$BIND_STATS_AIJ_BLKS_ PER_IO logical name allows you to override the default value of blocks per AIJ I/O. You can also set this threshold from the Configuration submenu in the Performance Monitor AIJ Analysis screen.

Legend:

T - True

F - False

E - Enabled

D - Disabled

Table 6–2 (Cont.) Predefined Variable Definitions

Variable Name	Type	Default Value	Min. Value	Max. Value	Scale	Description
AIJ_SEC_TO_EXTEND	Numeric	60	0		1	Specifies the default online analysis value of seconds to AIJ extend. The RDM$BIND_ STATS_AIJ_SEC_ TO_ EXTEND logical name allows you to override the default value of seconds to AIJ extend. You can also set this threshold from the Configuration submenu in the Performance Monitor AIJ Analysis screen.
ALARM	Numeric	0	0		0	Establishes an alarm interval (in seconds) for the Stall Messages screen from the command line. This is useful when you plan to submit the RMU Show Statistics command as a batch job. This variable supersedes the Alarm=seconds qualifier.
AUTO_ACTIVE_DETECT	Boolean	F	F	T		When the AUTO_NODE_DETECT variable is set to TRUE, this variable specifies whether to actively or passively detect new nodes joining the cluster. Active detection may incur an I/O operation per screen refresh. Passive detection relies on other users on the current node to passively refresh the node information.
AUTO_NODE_DETECT	Boolean	F	F	T		Specifies whether to automatically detect new nodes joining the cluster. When the utility detects a new node joining the cluster, the Cluster Statistics menu is automatically displayed.
AUTO_RECONNECT	Boolean	F	F	T		Specifies whether to automatically reconnect to disconnected nodes.

Legend:

T - True

F - False

E - Enabled

D - Disabled

Table 6–2 (Cont.) Predefined Variable Definitions

Variable Name	Type	Default Value	Min. Value	Max. Value	Scale	Description
BROADCAST	Boolean	T	F	T		Specifies whether to broadcast messages. The Broadcast qualifier is the default, if broadcasting of certain messages has been enabled with the DCL SET BROADCAST command. If broadcasting has been disabled with the DCL SET BROADCAST=NONE command, then broadcast messages are not displayed, even if you specify the RMU Show Statistics command with the Broadcast qualifier. This variable supersedes the Broadcast and Nobroadcast qualifiers.
BTR_FETCH_DUP_RATIO	Numeric	15	0		1	Specifies the default online analysis value of the B-tree duplicate fetch threshold. The RDM$BIND_STATS_BTR_LEF_ FETCH_RATIO logical name allows you to override the default value of the duplicate B-tree duplicate fetch threshold. You can also set this threshold from the Configuration submenu in the Performance Monitor Analysis screen.
BTR_LEF_FETCH_RATIO	Numeric	25	0		1	Specifies the default online analysis value of the B-tree leaf node fetch threshold. The RDM$BIND_STATS_BTR_LEF_ FETCH_RATIO logical name allows you to override the default value of the B-tree leaf node fetch threshold. You can also set this threshold from the Configuration submenu in the Performance Monitor Analysis screen.
CHECKPOINT_ALARM	Numeric	0	0		0	Establishes an alarm interval (in seconds) for the Checkpoint Information screen. This is useful when you plan to submit the RMU Show Statistics command as a batch job.
CHECKPOINT_SORT	String					Specifies the Checkpoint screen sort configuration options. The valid keywords are OLDEST_CHECKPOINT, OLDEST_ TRANSACTION, and OLDEST_QUIET_ POINT.

Legend:

T - True

F - False

E - Enabled

D - Disabled

Table 6–2 (Cont.) Predefined Variable Definitions

Variable Name	Type	Default Value	Min. Value	Max. Value	Scale	Description
CHECKPOINT_TX	String					Specifies the Checkpoint screen configuration options. The valid keywords are ACTUAL and ELAPSED.
CLUSTER_NODES	String					Identifies the set of nodes that are to participate in statistics collection for the current session. If more than one node is specified, they must be separated by commas. The keyword ALL_OPEN indicates that statistics should be collected from all nodes on which the database is currently open. Note that the ALL_OPEN keyword is never automatically generated. This variable supersedes the Cluster=node-list qualifier.
CONFIRM_EXIT	String	F				A value of TRUE indicates that you want to confirm before exiting the utility. A value of FALSE indicates you do not want to confirm before exiting the utility.
CUSTOM_LINE_n	String			18		Specifies the name of the statistic field located on line n of the Custom Statistics screen. Statistics may be entered for lines 1 through 36, although the number of lines that can actually be displayed depends on your terminal. The statistic name must be specified exactly as it appears on its home screen, including leading spaces. Duplicate statistics, as well as duplicated line numbers, are detected. Note that the specified custom statistics fields are not evaluated until after the database has been opened. Opening the database activates the various screens, which determines the set of custom statistic fields that can be specified. Therefore, some custom statistics fields may not always be available, depending on which database attributes (for instance, global buffers) are active.

Legend:

T - True

F - False

E - Enabled

D - Disabled

Table 6–2 (Cont.) Predefined Variable Definitions

Variable Name	Type	Default Value	Min. Value	Max. Value	Scale	Description
CYCLE	Numeric	0	0		0	Specifies the interval (in seconds) to automatically migrate to the next screen in the current submenu. This variable supersedes the CYCLE=seconds qualifier.
DASHBOARD_UPDATE	Boolean	D	D	E		Specifies whether or not dashboard updates are permitted (if you have the proper privileges). This variable supersedes the Option=Update and Option=Noupdate qualifiers.
DATABASE	Control					Identifies the database from which the configuration file was generated. This variable is for documentation purposes only, and is primarily useful only when the Log qualifier is specified.
DBKEY_LOG	String			255		Logs the records accessed during a given processing period by the various attached processes. The file-spec is the name of the file to which all accessed dbkeys are logged. When importing a configuration file that specifies a log file, even if the same log file is specified, a new log file is automatically created. This variable supersedes the Dbkey_ Log=dbkey-log qualifier.
DBR_RATIO	Numeric	15	0		1	Specifies the default online analysis value of the DBR invocation threshold. The RDM$BIND_STATS_DBR_RATIO logical name allows you to override the default value of the DBR invocation threshold. You can also set this threshold from the Configuration submenu in the Performance Monitor RUJ Analysis screen.
DEADLOCK_FULL_ DISPLAY	Boolean	D	D	E		Specifies whether or not the Lock Deadlock History screen is to display all processes or just those with deadlock messages.

Legend:

T - True

F - False

E - Enabled

D - Disabled

Table 6–2 (Cont.) Predefined Variable Definitions

Variable Name	Type	Default Value	Min. Value	Max. Value	Scale	Description
DEADLOCK_LOG	String			255		Specifies that lock deadlock messages are to be written to the specified file. This can be useful when you notice a great number of lock deadlock messages being generated, but do not have the resources to immediately investigate and resolve the problem. The file generated by the Deadlock_Log qualifier can be reviewed later so that the problem can be traced and resolved. When importing a configuration file that specifies a log file, even if the same log file is specified, a new log file is automatically created. This variable supersedes the Deadlock_Log=deadlock_log qualifier.
EVENT_DESCRIPTION	Command					This command describes a Show Statistics event and either enables a new event or disables an active event. Refer to Section 6.5 for more information. You can use the underscore character (_) or hyphen character (-) in place of spaces in statistic names that have leading spaces. For example, the statistic field name "file extend" can also be specified as "_ _ file_ extend" or "- - file-extend". This is useful for improving the readability of difficult statistic field names.
FULL_BACKUP_INTRVL	Numeric	6	0		0	Specifies the online analysis full database backup threshold. The RDM$BIND_STATS_FULL_ BACKUP_INTRVL logical name allows you to override the full database backup threshold. You can also set this threshold from the Configuration submenu in the Performance Monitor Recovery Analysis screen.

Legend:

T - True

F - False

E - Enabled

D - Disabled

Table 6–2 (Cont.) Predefined Variable Definitions

Variable Name	Type	Default Value	Min. Value	Max. Value	Scale	Description
GB_IO_SAVED_RATIO	Numeric	85	0		1	Specifies the online analysis GB IO-saved default threshold. The RDM$BIND_STATS_GB_IO_ SAVED_RATIO logical name allows you to override the GB IO-saved default threshold. You can also set the global buffer I/O-saved threshold from the Configuration submenu in the Performance Monitor Buffer Analysis screen.
GB_POOL_HIT_RATIO	Numeric	85	0		1	Specifies the online analysis GB pool hit default threshold. The RDM$BIND_STATS_GB_POOL_ HIT_RATIO logical name allows you to override the GB pool hit default threshold. You can also set the global buffer pool hit threshold from the Configuration submenu in the Performance Monitor Buffer Analysis screen.
HISTOGRAM	Boolean	F	F	T		Directs Oracle RMU to display the initial statistics screen in the numbers display mode or the graph display mode. The Histogram qualifier specifies the graph display mode. The Nohistogram qualifier specifies the numbers display mode. This variable supersedes the Histogram and Nohistogram qualifiers.
INCLUDE	Command					Temporarily switches processing to the quoted string value that defines a new configuration file. This configuration file, in turn, may also specify the INCLUDE command to switch to yet another configuration file. When processing of the command completes, processing of the current configuration file continues. Remember that variable definitions are superseded by subsequent references. The INCLUDE variable detects infinite recursion. The INCLUDE variable is not exported.

Legend:

T - True

F - False

E - Enabled

D - Disabled

Table 6–2 (Cont.) Predefined Variable Definitions

Variable Name	Type	Default Value	Min. Value	Max. Value	Scale	Description
INQUIRE	Command					Prompts the user to enter a value for the specified variable. This command is typically used after the PROMPT command to prompt the user to enter a value for certain variables. This command does not work when a configuration file is imported. The INQUIRE variable is not exported.
INTERACTIVE	Boolean	T	F	T		Displays the statistics dynamically to your terminal. The Interactive qualifier is the default when you execute the RMU Show Statistics command from a terminal. You can use the Nointeractive qualifier with the Output qualifier to generate a binary statistics file without generating a terminal display. The Nointeractive qualifier is the default when you execute the RMU Show Statistics command from a batch job. Note that most of these variables are not interesting when the INTERACTIVE variable is set to FALSE. This variable supersedes the Interactive and Nointeractive qualifiers.
LB_PAGE_HIT_RATIO	Numeric	75	0		1	Specifies the online analysis LB/AS page hit default threshold. The RDM$BIND_STATS_LB_PAGE_HIT_RATIO logical name allows you to override the LB/AS page hit default threshold. You can also set the local buffer pool hit threshold from the Configuration submenu in the Performance Monitor Buffer Analysis screen.
LOGICAL_AREA	String					Specifies the By Logical Area screen configuration options. The valid keywords are INDIVIDUAL and AGGREGATE.

Legend:

T - True

F - False

E - Enabled

D - Disabled

Table 6–2 (Cont.) Predefined Variable Definitions

Variable Name	Type	Default Value	Min. Value	Max. Value	Scale	Description
MAX_HASH_QUE_LEN	Numeric	2	0		0	Specifies the online analysis hash table queue length default threshold. The RDM$BIND_STATS_MAX_HASH_QUE_LEN logical name allows you to override the hash table queue length default threshold. You can also set the hash table queue length threshold from the Configuration submenu in the Performance Monitor Transaction Analysis screen.
MAX_LOCK_STALL	Numeric	2			1	Specifies the online analysis lock stall default threshold. The RDM$BIND_STATS_MAX_LOCK_STALL logical name allows you to override the lock stall default threshold. You can also set this threshold from the Configuration submenu in the Performance Monitor Locking Analysis screen.
MAX_TX_DURATION	Numeric	15	0		1	Specifies the online analysis transaction duration default threshold. The RDM$BIND_STATS_MAX_TX_DURATION logical name allows you to override the transaction duration default threshold. You can also set the transaction duration threshold from the Configuration submenu in the Performance Monitor Transaction Analysis screen.

Legend:

T - True

F - False

E - Enabled

D - Disabled

Table 6–2 (Cont.) Predefined Variable Definitions

Variable Name	Type	Default Value	Min. Value	Max. Value	Scale	Description
NOTIFY	String					Notifies the specified system operator or operators when a stall process exceeds the specified alarm interval by issuing a broadcast message and ringing a bell at the terminal receiving the message. The valid operator classes are: CENTRAL, CLUSTER, DISKS, SECURITY, and OPER1 through OPER12. Multiple operator classes can be separated by commas; for example ’OPER11,OPER12’. Be sure to use single quotation marks for this variable. This variable supersedes the Notify=oper-class qualifier.
OUTPUT	String			255		Specifies that the collected statistics are to be written to the specified binary output file. This variable supersedes the Output=binary-file qualifier.
PAGES_CHECKED_ RATIO	Numeric	10	0		1	Specifies the online analysis pages checked default threshold. The RDM$BIND_STATS_PAGES_ CHECKED_RATIO logical name allows you to override the pages checked default threshold. You can also set this threshold from the Configuration submenu in the Performance Monitor Record Analysis screen.
PRINT	Command					Prints the value to the log file. This variable is useful if you do not want to use the Log qualifier, but want to display the value of a variable. This variable is also useful for displaying the initial value of a variable before it is changed in the configuration file. The PRINT variable is not exported.

Legend:

T - True

F - False

E - Enabled

D - Disabled

Table 6–2 (Cont.) Predefined Variable Definitions

Variable Name	Type	Default Value	Min. Value	Max. Value	Scale	Description
PROMPT	Command					Prints the value to the terminal. This variable is typically used in conjunction with the INQUIRE variable, but can also be useful if you do not want to use the Log qualifier, but want to display the value of a variable. This command does not work when a configuration file is imported. The PROMPT variable is not exported.
PROMPT_TIMEOUT	Integer	60	10			Allows you to specify the user prompt timeout interval (in seconds). If you specify a value of 0, no user prompts are timed out. Specifying a value of 0 can cause a database hang situation. Therefore, Oracle Corporation recommends that you do not specify a value of 0 unless you are certain that prompts will always be responded to in a timely manner.
RECS_FETCHED_RATIO	Numeric	20	0		1	Specifies the online analysis records fetched default threshold. The RDM$BIND_STATS_RECS_FETCHED_RATIO logical name allows you to override the records fetched default threshold. You can also set this threshold from the Configuration submenu in the Performance Monitor Record Analysis screen.
RECS_STORED_RATIO	Numeric	20	0		1	Specifies the online analysis records stored default threshold. The RDM$BIND_STATS_RECS_STORED_RATIO logical name allows you to override the records stored default threshold. You can also set this threshold from the Configuration submenu in the Performance Monitor Record Analysis screen.

Legend:

T - True

F - False

E - Enabled

D - Disabled

Table 6–2 (Cont.) Predefined Variable Definitions

Variable Name	Type	Default Value	Min. Value	Max. Value	Scale	Description
REDIRECT	Command					Forces the redirection of another variable's value into itself. This is the means by which variable indirection can be achieved. Command variables and control variables cannot be redirected, but all other variables can be. For example, if the variable HELLO has the value THERE, and the variable FOO has the value HELLO, then the command REDIRECT="FOO" causes the variable FOO to have the value THERE. This command variable is typically used after defining the variable's value using the INQUIRE command. The REDIRECT variable is not exported.
REFRESH_INTERVAL	Numeric	3	0		0	Specifies the statistics collection interval in seconds. If you omit this qualifier, a sample collection is made every 3 seconds. The integer has a normal range of 1 to 180 (1 second to 3 minutes). However, if you specify a negative number for the Time qualifier, the RMU Show Statistics command interprets the number as hundredths of a second. For example, Time=20 specifies an interval of 20/100 or 1/5 of a second. This variable supersedes the Time=seconds qualifier.
REOPEN_INTERVAL	Numeric	0	0		0	After the specified interval, closes the current output file and opens a new output file without requiring you to exit from the Performance Monitor. The new output file has the same name as the previous output file, but the version number is incremented by 1. This variable supersedes the Reopen_Interval=seconds qualifier.

Legend:

T - True

F - False

E - Enabled

D - Disabled

Table 6–2 (Cont.) Predefined Variable Definitions

Variable Name	Type	Default Value	Min. Value	Max. Value	Scale	Description
RESET	Boolean	F	F	T		Specifies whether or not the statistics are to be automatically reset prior to being displayed. This variable is always exported with the value FALSE regardless of its initial value. This variable supersedes the Reset and Noreset qualifiers.
RUJ_SYNCH_IO_RATIO	Numeric	10	0		1	Specifies the online analysis synchronous RUJ I/O default threshold. The RDM$BIND_STATS_RUJ_SYNC_IO_RATIO logical name allows you to override the synchronous RUJ I/O default threshold. You can also set this threshold from the Configuration submenu in the Performance Monitor RUJ Analysis screen.
SCREEN	String			255		Specifies the name of the initial screen to be displayed. This variable supersedes the Screen=screen_name qualifier.
STALL_LOG	String			255		Specifies that stall message are to be written to the specified file. This can be useful when you notice a great number of stall messages being generated, but do not have the resources to immediately investigate and resolve the problem. The file generated by the Stall_Log qualifier can be reviewed later so that the problem can be traced and resolved. When you import a configuration file that specifies a log file, even if the same log file is specified, a new log file will be automatically created. This variable supersedes the Stall_Log=file-spec qualifier

Legend:

T - True

F - False

E - Enabled

D - Disabled

Table 6–2 (Cont.) Predefined Variable Definitions

Variable Name	Type	Default Value	Min. Value	Max. Value	Scale	Description
STALL_LOG_VERBOSE	String					Specifies whether the stall messages logging facility will report a stall message at each specified interval, even if it has previously been reported. The valid keywords are ENABLED and DISABLED.
STALL_MESSAGE	String			255		Specifies the Stall Messages screen configuration options. The valid keywords are ACTUAL and ELAPSED.
SYSTEM_LOGICAL_ AREAS	Boolean	T	F	T		Specifies whether or not to display system relations on the Logical Area statistics screens.
TIMEOUT_FULL_ DISPLAY	Boolean	D	D	E		Specifies whether or not the Lock Timeout History screen is to display all processes or just those with deadlock messages.
TIMEOUT_LOG	String			255		Specifies that lock timeout messages are to be written to the specified file. This can be useful when you notice a great number of lock timeout messages being generated, but do not have the resources to immediately investigate and resolve the problem. The file generated by the Timeout_Log qualifier can be reviewed later so that the problem can be traced and resolved. When you import a configuration file that specifies a log file, even if the same log file is specified, a new log file will be automatically created. This variable supersedes the Timeout_ Log=timeout_log qualifier

Legend:

T - True

F - False

E - Enabled

D - Disabled

Table 6–2 (Cont.) Predefined Variable Definitions

Variable Name	Type	Default Value	Min. Value	Max. Value	Scale	Description
TX_DURATION	String					Specifies the Tx Duration screen configuration options. The valid keywords are TOTAL, READ_WRITE, and READ_ONLY.
UNSET	Command					This command removes a user-defined variable from the symbol table. It is not necessary to unset a user-defined variable prior to changing its value. It is only necessary to unset a variable if you do not wish it to exist for some reason. Note that using the single quotation mark (') is not supported for this command because variables can be specified using double quotation marks (") only. The UNSET variable is not exported.
VERB_SUCCESS_RATIO	Numeric	25	0		1	Specifies the online analysis verb success default threshold. The RDM$BIND_STATS_VERB_SUCCESS_RATIO logical name allows you to override the verb success default threshold. You can also set the verb success threshold from the Configuration submenu in the Performance Monitor Transaction Analysis screen.

Legend:
T - True
F - False
E - Enabled
D - Disabled

6.6.13 Example of a Poorly Constructed Configuration File

The following is an example of a poorly designed configuration file. It shows how a misunderstanding of how quotation marks are applied can lead to unexpected results. It is presented here to provide examples of what *not* to do.

```
PRINT="DATABASE";              # Displays the name of the database
PRINT='DATABASE';              # Displays the word "database"
UNSET="FOO";                   # Returns warning: unknown variable "FOO"
FOO=5;
UNSET='FOO';                   # Returns warning: unknown variable "FOO"
UNSET="FOO";
PRINT="FOO";                   # Displays the word "FOO", not the value 5
```

```
123=456;                      # Returns warning: incorrect variable name "123"
PRINT="PRINT";                # Displays the word "PRINT"
INCLUDE="INCLUDE";            # Returns error: %COSI-E-FNF, file not found
INCLUDE="CONFIG.CFG";         # Returns error: %COSI-F-EXIST, file exists
CUSTOM_LINE_5="transactions   # Returns error warning: unterminated string
CUSTOM_LINE_5="transactions";
CUSTOM_LINE_5="verb successes";        #  Returns warning: replacing defined variable "CUSTOM_LINE_5"
CUSTOM_LINE_6="verb successes";        #  Returns warning: replacing defined statistic "transactions"
CUSTOM_LINE_7="verb failures";
CUSTOM_LINE_8="verb confusion";        # Returns warning: custom statistic field "verb confusion" not found.
HERE="there";
PRINT="here and now";         # Displays the words "here and now"
PRINT="here";                 # Displays the word "there"
PRINT='here';                 # Displays the word "here"
YES="yes";
PRINT="yes";                  # Displays the phrase "YES = yes"
PRINT = "CYCLE";              # Displays the phrase "CYCLE = 0"
CYCLE = "5";
CYCLE = 5;
PRINT = "CYCLE";              # Displays the phrase "CYCLE = 5"
CYCLE = ALARM;
PRINT = "CYCLE";              # Displays the phrase "CYCLE = 0"
NO="no";
PRINT="no";                   # Displays the phrase "NO = no"
BROADCAST = BROADCAST;        # Returns warning: invalid value "BROADCAST"
NOTIFY = "CLUSTER";           # Displays the phrase "NOTIFY = ''"
```

6.7 Cluster Statistics Collection and Presentation

The RMU Show Statistics command has been enhanced with the Cluster Statistics Collection and Presentation feature. The purpose of this feature is to allow real-time collection and presentation of database statistics from other nodes within the cluster where the database is currently being accessed.

The Cluster Statistics Collection and Presentation feature is invoked using the RMU Show Statistics command with the Cluster qualifier. This qualifier specifies the list of remote nodes from which statistics collection and presentation is to be performed. The collected statistics are merged with the current node information and displayed using the usual statistics screens.

The following list shows the use of the Cluster qualifier:

- If the Cluster qualifier is specified by itself, then remote statistics collection is performed on all cluster nodes on which the database is currently open.

- If the Cluster=(node-list) qualifier is specified, then remote statistics collection is performed on the specified nodes only, even if the database is not yet open on one or more of those nodes.
- If the Cluster qualifier is not specified, or the Nocluster qualifier (the default) is specified, then cluster statistics collection is not performed. However, you may still enable clusterwide statistics collection online using the Tools menu.

Up to 95 different cluster nodes can be specified using the Cluster qualifier. There is a maximum number of 95 cluster nodes because Oracle Rdb supports a maximum of 96 nodes per database. The current node is always included in the list of nodes from which statistics collection is to be performed.

It is not necessary to have the RMU Show Statistics command running on the specified remote nodes or to have the database open on the remote nodes. These events are automatically handled by the feature.

The following example shows the use of the Cluster qualifier to initiate statistics collection and presentation from two remote nodes:

```
$ RMU /SHOW STATISTICS /CLUSTER=(BONZAI, ALPHA4) MF_PERSONNEL
```

Remote nodes can also be added and removed online at run time. Use the Cluster Statistics option located in the Tools menu. The Tools menu is displayed by using the exclamation point (!) on-screen menu option.

The Cluster Statistics menu contains up to six options, depending on the current state of remote statistics collection. These menu options are the following:

- Manually add remote node to collection - Allows you to type in the name of a remote node for which you want to start statistics collection. Up to 95 different remote nodes can be added using this option.

  This option is useful for entering a comma-separated list of cluster nodes to be connected. Alternatively, the node name specified can be a logical name whose definition is a comma-separated list of cluster nodes to be connected. When defining the logical name, be sure to place the comma-separated cluster list in double quotation marks so that it is processed as a single logical name and not a search-list logical name. The following defines a logical name containing the names of three cluster nodes:

  ```
  $ DEFINE NODENAME "BONZAI,ALPHA4,ALPHA5"
  ```

  When a list of cluster nodes is specified, connection to the nodes is terminated upon encountering any error condition. Any previously connected nodes remain connected. You are notified when an error condition is encountered.

Note that you cannot add the current node to the cluster statistics collection list because statistics are always collected on the current node.

- Select cluster node for collection - Displays an alphabetic menu of all available cluster nodes in the cluster and lets you select the cluster node for which you want to start statistics collection. Up to 95 different cluster nodes can be added using this option.
- Remove cluster node from collection - Displays a menu of cluster nodes that are currently activated with the Cluster Statistics Collection and Presentation feature. Selecting one of these nodes will stop cluster statistics collection from that node. This option is only available when cluster statistics are being actively collected.

  Note that you cannot remove the current node from the cluster statistics collection list.
- Display active collection nodes - Displays a menu of cluster nodes that are currently activated with the Cluster Statistics Collection and Presentation feature. This option is available for your information. This option always displays the current node first, in case you forget which node you are using.

  Each line of the display lists the node name, the date when the database was opened on that node, and whether cluster statistics collection is active or inactive. A cluster node becomes inactive when a network transmission error occurs (see the description of the Reconnect to inactive cluster nodes menu option that appears later in this list).
- Display available cluster nodes - Displays an alphabetic list of all valid cluster nodes in your cluster. This option is available for your information.
- Reconnect to inactive cluster nodes - Displayed when there are one or more inactive cluster nodes in your cluster statistics collection (see the description of the Display active collection nodes menu option that appears previously in this list). This option is used to attempt to reconnect to the cluster node manually; if it is unsuccessful, the cluster node is automatically removed from cluster statistics collection.

Note: Adding a new node or deleting an existing node from the cluster statistics collection automatically resets all of the statistics information so that you have a consistent point-in-time view of the statistics information. If desired, you can unreset the statistics using the Unreset on-screen menu option.

The following example shows a sample Active User Stall Messages screen:

```
Node: Cluster (2)    Oracle Rdb V7.1 Performance Monitor 16-DEC-1997 13:03:06
Rate: 1.00 Second          Active User Stall Messages       Elapsed: 00:03:32.35
Page: 1 of 1   USER1:[SPANDERSON.WORK.STATS]MF_PERSONNEL.RDB;1     Mode: Online
--------------------------------------------------------------------------------
Process.ID Elapsed.... T Stall.reason............................. Lock.ID.
20600CB8:1u              reading pages 1:667 to 1:667
20600CB5:1*00:00:16.55 W waiting for record 51:60:2 (EX)
20600CB6:1*
232068CC:1u              reading ROOT file (RTUPB VBN 98)
20608403:1R              binding to database
--------------------------------------------------------------------------------
Config Exit Help LockID Menu >next_page <prev_page Set_rate Write Zoom !
```

Note that the Node section of the header region indicates a cluster and identifies the total number of nodes from which information is being collected, including the current node. On screens for which remote information is not collected, the node name indicates only the current node name.

On the Stall Messages and Active User Stall Messages screens, the processes from cluster nodes are highlighted to permit easy identification. Also, cluster application processes are suffixed with an asterisk (*) for easy identification when the screen is written to disk.

The following example shows a sample Summary IO Statistics screen:

```
Node: Cluster (2)    Oracle Rdb V7.1 Performance Monitor 17-DEC-1997 06:49:23
Rate: 1.00 Second            Summary IO Statistics          Elapsed: 00:02:20.28
Page: 1 of 1    USER1:[SPANDERSON.WORK.STATS]MF_PERSONNEL.RDB;1     Mode: Online
--------------------------------------------------------------------------------
statistic.........      rate.per.second............. total....... average......
name..............      max..... cur..... avg....... count....... per.trans....
transactions                   0        0        0.6           98           1.0
verb successes                 0        0        9.9         1397          14.2
verb failures                  0        0        0.7          100           1.0
synch data reads               0        0       12.4         1751          17.8
synch data writes              0        0        4.7          669           6.8
asynch data reads              0        0        3.9          549           5.6
asynch data writes             0        0        0.3           44           0.4
RUJ file reads                 0        0        0.0            2           0.0
RUJ file writes                0        0        0.0           10           0.1
AIJ file reads                 0        0        0.5           74           0.7
AIJ file writes                0        0        0.7          111           1.1
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      1.6       227         2.3
root file writes               0      0      0.6        95         0.9
--------------------------------------------------------------------------------
Exit Graph Help Menu Options Pause Reset Set_rate Time_plot Unreset Write X_plot
```

Note that, other than the Cluster designation in the screen header region, ordinary numeric statistics screens appear exactly as they usually do.

While the main set of numeric statistics (such as the summary statistics screens) are always collected, the remote collection of by-cache, by-area, and by-logical-area statistics is based upon the current screen being displayed. For example, the per-process stall information is collected only when one of the stall messages screens is being displayed. In addition, not all screens have remotely collected information; you will be notified when you migrate to a screen that does not support remote statistics collection. However, the majority of statistics screens do support remote statistics collection.

Note: The per-screen statistics collection feature is designed to minimize network use and improve overall system performance. However, this feature may affect the writing of by-area data to the binary output file when the Output qualifier is specified. For example, because the by-area cluster statistics are only collected when a by-area screen is being displayed, by-area information written to the binary output file may be out-of-date if the current screen is not a by-area statistics screen.

The following caveats apply to the Cluster Statistics Collection and Presentation feature:

- Up to 95 cluster nodes can be specified. However, use cluster statistics collection prudently, as the system overhead in collecting the remote statistics may be substantial, depending on the amount of information being transmitted on the network.
- Cluster statistics are collected at the specified display refresh rate. Therefore, the display refresh rate should be set to a reasonable rate based on the number of cluster nodes being collected. The default refresh rate of 3 seconds is reasonable for most remote collection loads.
- If the Cluster qualifier is specified, the list of cluster nodes will apply to any database accessed during the Show Statistics session. When accessing additional databases using the Switch Database option, the same cluster nodes

will be automatically accessed. However, any nodes manually added using the Cluster Statistics menu will not automatically be added to the remote collection of the new database.

In other words, manually adding and deleting cluster nodes affects only the current database and does not apply to any other database that may have been accessed during the session. For example, when you run the Show Statistics utility on node ALPHA3 with manually added node BONZAI, subsequently switching to BONZAI as the current node will not display cluster statistics from node ALPHA3 unless you manually add that node. Furthermore, switching back to node ALPHA3 as the current node loses the previous collection of node BONZAI because it was manually added.

- Both DECnet and TCP/IP network protocols are supported. By default, the DECnet protocol is used. To explicitly specify which network protocol to use, define RDM$BIND_STT_NETWORK_TRANSPORT to DECNET or TCPIP respectively. The RDM$BIND_STT_NETWORK_TRANSPORT logical name must be defined to the same definition on both the local and the cluster nodes. The RDM$BIND_STT_NETWORK_TRANSPORT logical name can be specified in LNM$FILE_DEV on the local node, but must be specified in the LNM$SYSTEM_TABLE on all remote nodes.

Note: There is no command qualifier to specify the network protocol.

- The Output qualifier continues to work as usual, but when in cluster mode writes the cluster statistics information to the binary output file.
- The Cluster qualifier cannot be specified with the Input qualifier. Furthermore, the online selection of cluster nodes is not available when you use the Input qualifier.
- While the Cluster Statistics Collection and Presentation feature is active, all on-screen menu options continue to operate as usual. This includes the time-plot, scatter-plot, screen pause, and various other options.
- There is no way to exclude the current node from statistics collection. Log in to another node if you want to do this.
- The cluster collection of per-process stall information automatically detects the binding or unbinding of processes to cluster databases. There is no need to manually refresh the database information on the current node.

- If the database is not currently open on the specified node, cluster statistics collection will still be attempted. However, the remote database must still be opened normally prior to regular process attaches.
- An attempt to add more than 95 cluster nodes results in an error.
- When you display any of the per-process screens that support cluster statistics collection, such as the Stall Messages screen, zooming on any of the displayed processes will display the node that process is using.
- Using the Cluster Statistics submenu from the Tools menu, it is also possible to collect statistics from all open database nodes using the "Collect from open database nodes" menu option. This option simplifies the DBA's job of remembering where the database is currently open. However, subsequently opened nodes will not be automatically added to the collection; these must be added manually.
- The cluster statistics collection is an *intracluster* feature in that it works only on the same database, using the same device and directory specification used to run the initial RMU Show Statistics command (that is, on a shared disk). The cluster statistics collection does not work across clusters (*intercluster*).
- When you replay a binary output file, the screen header region accurately reflects the number of cluster nodes whose statistics are represented in the output file.

6.7.1 Changes to Screen Header Region

The introduction of clusterwide statistics collection for a database has made it important to know when clusterwide statistics are being collected. Equally important, however, is knowing whether statistics are being collected on all available cluster nodes on which the database is being accessed.

The following example shows the original RMU Show Statistics utility screen, with the standard three-line header region.

```
Node: ALPHA3   Oracle Rdb V7.1 Performance Monitor 27-MAR-1997 07:25:57.96
Rate: 1.00 Second           Summary IO Statistics         Elapsed: 00:00:03.27
Page: 1 of 1    USER1:[SPANDERSON.WORK.STATS]MF_PERSONNEL.RDB;1   Mode: Online
--------------------------------------------------------------------------------
statistic.........      rate.per.second............. total....... average......
name..............      max..... cur..... avg....... count....... per.trans....
transactions                   0        0        0.0            0           0.0
verb successes                 0        0        0.0            0           0.0
verb failures                  0        0        0.0            0           0.0
```

```
synch data reads              0        0       0.9           3          0.0
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               5        5       5.8          19          0.0
root file writes              0        0       0.0           0          0.0
--------------------------------------------------------------------------------
Exit Graph Help Menu Options Pause Reset Set_rate Time_plot Write X_plot Yank !
```

It is difficult to know on how many nodes the database is currently active, as well as the maximum number of nodes on which the database may be active.

The RMU Show Statistics screen header region has been enhanced to display the current node suffixed with an x/y/z identifier. The value x indicates the number of nodes on which statistics are being actively collected. The value y indicates the number of nodes on which the database is being accessed. The value z indicates the maximum number of nodes on which the database can be accessed.

For example, the indicator 1/3/5 specifies that statistics are being collected from one node, but that the database is currently open on three nodes, with a maximum open node count of five.

The following example demonstrates this:

```
Node: ALPHA3 (1/3/5)    Oracle Rdb V7.1 Perf. Monitor 27-MAR-1997 07:36:56.08
Rate: 1.00 Second            Summary IO Statistics          Elapsed: 00:00:02.20
Page: 1 of 1    USER1:[SPANDERSON.WORK.STATS]MF_PERSONNEL.RDB;1    Mode: Online
--------------------------------------------------------------------------------
statistic.........      rate.per.second............ total....... average......
name..............      max..... cur..... avg....... count....... per.trans....
transactions                  0        0       0.0           0          0.0
verb successes                0        0       0.0           0          0.0
verb failures                 0        0       0.0           0          0.0
synch data reads              1        1       1.3           3          0.0
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              9        9        9.0           20           0.0
root file writes             0        0        0.0            0           0.0
--------------------------------------------------------------------------------
Exit Graph Help Menu Options Pause Reset Set_rate Time_plot Write X_plot Yank !
```

When statistics are being collected from more than one node, the Node tag is changed to Cluster in the header region to designate that clusterwide statistics collection is active.

For example, the following screen shows that statistics are being collected from three nodes:

```
Cluster: ALPHA3 (3/3/5) Oracle Rdb V7.1 Perf. Monitor 27-MAR-1997 07:37:13.33
Rate: 1.00 Second            Summary IO Statistics          Elapsed: 00:00:02.86
Page: 1 of 1    USER1:[SPANDERSON.WORK.STATS]MF_PERSONNEL.RDB;1    Mode: Online
--------------------------------------------------------------------------------
statistic.........      rate.per.second............. total....... average......
name..............      max..... cur..... avg....... count....... per.trans....
transactions                 0        0        0.0            0           0.0
verb successes               0        0        0.0            0           0.0
verb failures                0        0        0.0            0           0.0
synch data reads             0        0        0.0            0           0.0
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             10        0        1.0            3           0.0
root file writes             0        0        0.0            0           0.0
--------------------------------------------------------------------------------
Exit Graph Help Menu Options Pause Reset Set_rate Time_plot Unreset Write X_plot
```

6.8 Automatic Screen Capture Facility

The RMU Show Statistics command provides an Automatic Screen Capture facility. This facility allows you to automatically capture images of all screens at a specified interval. The facility is similar to intermittently using the Options on-screen menu option.

You invoke the Automatic Screen Capture facility with the "Start automatic screen capture" option on the Tools menu (by entering the exclamation point (!)). You will be asked to enter the interval between screen capture operations, expressed in seconds. The minimum interval is 30 seconds.

It takes approximately 5 to 10 seconds for RMU to capture all available screens. You will be notified with the message, ***Writing Report***, in the status region of the current screen when the screens are being captured.

To guarantee consistent statistical information, statistical information updates are temporarily paused while the screen capture operation is occurring. This pause also affects writing to the binary output file, as well as any log files being recorded.

You disable the Automatic Screen Capture facility with the "Stop automatic screen capture" option on the Tools menu.

You can also invoke the Automatic Screen Capture facility with the configuration REPORT_INTERVAL variable specifying the number of seconds.

There is no command qualifier for this facility. Also, you cannot use the facility if the Nointeractive qualifier is specified.

The Automatic Screen Capture facility works with binary files.

The Automatic Screen Capture facility is integrated with the Cluster Statistic Collection facility. If cluster statistics collection is enabled, all supported screens will provide cluster information.

6.9 New Display of AIJ Backup Activity

The RMU Show Statistics command has been enhanced with the new AIJ Backup Activity screen. Located in the Process Information submenu, the AIJ Backup Activity screen displays information about each AIJ backup operation being performed on the node.

On OpenVMS, the AIJ Backup Activity screen is also available during clusterwide statistics collection. This means you can monitor the activities of all AIJ backup operations occurring on any node accessing the database.

The AIJ Backup Activity screen information is not recorded in the binary output file. Therefore, the screen is not available during binary file replay.

The following example shows a sample AIJ Backup Activity screen:

```
Node: ALPH   (1/1/16)   Oracle Rdb V7.1 Perf. Monitor 19-DEC-1997 15:50:24.49
Rate: 0.50 Seconds           AIJ Backup Activity          Elapsed: 00:03:57.99
Page: 1 of 1             DISK$:[WORK]MF_PERSONNEL.RDB;1     Mode: Online
--------------------------------------------------------------------------------
Process.ID  Activity... VBN...... Operation......................... Lock.ID.
34218467:1s block bkup  7:1017    Initializing AIJ journal
--------------------------------------------------------------------------------
```

The following example shows the same AIJ backup operation in a later stage of the backup operation:

```
Node: ALPH   (1/1/16)   Oracle Rdb V7.1 Perf. Monitor 19-DEC-1997 15:50:24.99
Rate: 0.50 Seconds           AIJ Backup Activity          Elapsed: 00:03:58.49
Page: 1 of 1             DISK$:[WORK]MF_PERSONNEL.RDB;1     Mode: Online
--------------------------------------------------------------------------------
Process.ID  Activity... VBN...... Operation......................... Lock.ID.
34218467:1s finish      7:1017    writing ROOT file (AIJFB VBN 1228)


--------------------------------------------------------------------------------
```

The AIJ Backup Activity screen contains five columns of information:

- Process.ID: This field contains the process identifier of the AIJ backup process. This process can be the AIJ backup server (ABS), in which case the process identifier contains the "s" suffix. This process can also be the manual RMU Backup After_Journal command, in which case the process identifier contains the "u" suffix.

  Additional information can be obtained about this process by using the Zoom on-screen menu option.

- Activity: This field contains a description of the backup activity being performed by the AIJ backup utility. The following backup activities are displayed:
  - activation - The AIJ backup utility is being invoked by the monitor if it is the ABS, or startup if the manual backup utility is being used.
  - bind - The AIJ backup utility is binding to the database.
  - start - The AIJ backup utility is starting the backup operation.
  - create bkup - The AIJ backup utility is creating a disk-based backup file.
  - create temp - The AIJ backup utility is creating a temporary AIJ journal. This activity typically occurs when the fast commit feature is used in conjunction with extensible AIJ journals.

  - record bkup - The AIJ backup utility is backing up an extensible AIJ journal to disk, or any type of AIJ journal to tape, using a record-by-record transfer algorithm.
  - block bkup - The AIJ backup utility is backing up a fixed-size (circular) AIJ journal to disk, using a 127-block transfer algorithm.
  - finish - The AIJ backup utility is completing the backup of an AIJ journal.
  - quiet-point - The AIJ backup utility is attempting to acquire the quiet-point lock.
  - record shfl - The AIJ backup utility is performing the record shuffle operation used for extensible AIJ journals.
  - unbind - The AIJ backup utility is unbinding from the database.
- VBN - This column identifies the current block number of the AIJ journal being backed up. The block number is normally prefixed with the AIJ sequence number, so it is easy to identify which AIJ journal is being backed up.
- Operation -This column identifies the activity-specific operation being performed by the AIJ backup utility. This column contains messages similar to those displayed by the Stall Messages screen.
- Lock.ID - This column identifies any lock the AIJ backup utility may be trying to acquire. This lock is typically the quiet-point lock.

  Use the LockID on-screen menu option to obtain more information about this lock.

6.10 Page Information Zoom Screen

The RMU Statistics command has been integrated with the RMU Dump command to provide run-time database page information displayed on a zoom screen. The page information is presented on a zoom screen in a format similar to that displayed by the RMU Dump command with the Area, Start, and End qualifiers.

The page information zoom screen is available from the Stall Messages, Active User Stall Messages, DBR Activity, and DBKEY Information screens.

Select page information on the PageInfo on-screen menu option by pressing the P key.

RMU prompts you to select a process from the list of available processes with dbkey information displayed on the screen. Only those processes displaying physical dbkey information can be selected.

If the process you select is accessing a range of pages, you are prompted to select the desired page from a submenu provided.

If you are displaying page information from the DBKEY Information page, you are prompted to select one of the types of pages being accessed by that process.

You can also display an arbitrary page using the Tools menu, by entering the exclamation point (!). If you select the Display Page Information option, you are prompted for the desired storage area and page number.

The following caveats apply to the page information display:

- For security reasons, you cannot display the contents of individual lines on a data page or the contents of area inventory pages. Contact your database administrator for other methods to display the contents of selected rows.
- The page information can be quite lengthy. You can migrate through the various pages using the right-arrow and left-arrow keys (1 page at a time) or the up-arrow and down-arrow keys (1 line at a time).
- The page information zoom screen contents can be written to disk using the Write on-screen menu option (W key).
- The PageInfo on-screen menu option is not available during replay of a binary input file.
- Because no locking of the selected page actually occurs, it is possible (but unlikely) to display inconsistent page information.

The PageInfo on-screen menu option identifies and resolves logical dbkeys and retrieves the corresponding physical dbkeys.

Note: When Oracle Rdb uses ALG, logical dbkeys such as "59:1:-3" are not resolvable, so the RMU Show Statistics command retrieves the identified page, which is not always correct. In the preceding example, page "1" is a SPAM page, which cannot be the target of the logical dbkey.

The following is an example of a live data page information display:

```
+-----------------------------------------------------------------------+
|                    0001 00000005  0000  page 5, physical area 1 (data)        |
|                         8EE86A99  0006  checksum = 8EE86A99                     |
|                009BA463 0DA1FC74  000A  time stamp = 14-SEP-1997 06:51:12.75 |
|                        0000 006A  0012  106 free bytes, 0 locked             |
```

```
|                        0002  0016  2 lines                                  |
|                   01AE 0240  0018  line 0: offset 0240, 430 bytes            |
|                   01AE 0092  001C  line 1: offset 0092, 430 bytes            |
|                    0000018C  0020  line 0: TSN 396                           |
|                    000001BF  0024  line 1: TSN 447                           |
|                    00000024  03EE  snap page pointer 36                      |
|                    000001BF  03F2  snap pointer TSN 447                      |
|                        003B  03F6  logical area 59                           |
|                    0000003B  03F8  page sequence number 59                   |
|                        0000  03FC  page TSN base 0                           |
|                        0000  03FE  MBZ '..'                                  |
+-----------------------------------------------------------------------------+
```

The following is an example of a snapshot data page information display:

```
+-----------------------------------------------------------------------------+
|                4001 00000001  0000  page 1, physical area 1 (snap)          |
|                     A46ACD6A  0006  checksum = A46ACD6A                      |
|            009BA304 993A8B26  000A  time stamp = 12-SEP-1997 13:02:33.60     |
|                    0000 0054  0012  84 free bytes, 0 locked                  |
|                         0003  0016  3 lines                                  |
|                    0000 0000  0018  line 0: empty                            |
|                    01AE 0238  001C  line 1: offset 0238, 430 bytes           |
|                    01AE 008A  0020  line 2: offset 008A, 430 bytes           |
|                     00000000  0024  line 0: TSN 0                            |
|                     000085BD  0028  line 1: TSN 34237                        |
|                     000085BF  002C  line 2: TSN 34239                        |
|                         0000  0030  line 0 -> live line: 0                   |
|                         0000  0032  line 1 -> live line: 0                   |
|                         0000  0034  line 2 -> live line: 0                   |
|                     000001AB  03E6  live page pointer 427                    |
|                     000085C4  03EA  max TSN 34244                            |
|                     FFFFFFFF  03EE  snap page pointer -1                     |
|                     00000000  03F2  snap pointer TSN 0                       |
|                         0000  03F6  MBZ '..'                                 |
|                     00000000  03F8  page sequence number 0                   |
|                         0000  03FC  page TSN base 0                          |
|                         0000  03FE  MBZ '..'                                 |
+-----------------------------------------------------------------------------+
```

The following is an example of an area inventory page (AIP) information display:

```
+-----------------------------------------------------------------------------+
|                0001 000001D0  0000  page 464, physical area 1 (AIP)         |
|                     D992664E  0006  checksum = D992664E                      |
|            009646FF 8BFE44E0  000A  time stamp =  1-DEC-1992 12:49:48.       |
|                    0000 0022  0012  34 free bytes, 0 locked                  |
```

```
|                     000001D1  0016  next area inventory page 465          |
|                         4001  03F6  logical area 16385                     |
|                     00000000  03F8  page sequence number 0                 |
|                         0000  03FC  page TSN base 0                        |
+---------------------------------------------------------------------------+
```

The following is an example of a SPAM page information display; note that a SPAM page display is quite lengthy:

```
+---------------------------------------------------------------------------+
|                    0001 00000001  0000  page 1, physical area 1 (SPAM)      |
|                         4681E156  0006  checksum = 4681E156                |
|                80000000 00000060  000A  Fast incremental backup TSN = 0:96  |
|                        0000 0001  0012  1 free byte, 0 locked              |
| FFFFFFFFFFFFFFFCFFFFFFFFFFFFFFFF  0016  pages 2-31: threshold 3             |
|                                         page 32: threshold 0                |
|                                         pages 33-65: threshold 3            |
| FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF  0026  pages 66-129: threshold 3           |
| FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF  0036  pages 130-193: threshold 3          |
| FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF  0046  pages 194-257: threshold 3          |
| FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF  0056  pages 258-321: threshold 3          |
| 0FFFFFFC33C3FFFFFFFFFFFFFFFFFFFF  0066  pages 322-362: threshold 3          |
|                                         pages 363-364: threshold 0          |
|                                         pages 365-366: threshold 3          |
|                                         page 367: threshold 0               |
|                                         page 368: threshold 3               |
|                                         pages 369-370: threshold 0          |
|                                         pages 371-383: threshold 3          |
|                                         pages 384-385: threshold 0          |
| 33FFFFFF03FFFFFC03FFFFF3FF0FFFFF  0076  pages 386-395: threshold 3          |
|                                         pages 396-397: threshold 0          |
|                                         pages 398-402: threshold 3          |
|                                         page 403: threshold 0               |
|                                         pages 404-414: threshold 3          |
|                                         pages 415-418: threshold 0          |
|                                         pages 419-430: threshold 3          |
|                                         pages 431-433: threshold 0          |
|                                         pages 434-446: threshold 3          |
|                                         page 447: threshold 0               |
|                                         page 448: threshold 3               |
|                                         page 449: threshold 0               |
|                                         page 513: threshold 0               |
|                                                 .                           |
|                                                 .                           |
|                                                 .                           |
```

```
|                             0052  03E7  pages 1055-1057, logical area 82     |
|                             0052  03E9  pages 1058-1060, logical area 82     |
|                             0052  03EB  pages 1061-1063, logical area 82     |
|                             0052  03ED  pages 1064-1066, logical area 82     |
|                             0052  03EF  pages 1067-1069, logical area 82     |
|                             0052  03F1  pages 1070-1072, logical area 82     |
|                             0052  03F3  pages 1073-1075, logical area 82     |
|                             0052  03F5  pages 1076-1078, logical area 82     |
|                             0052  03F7  pages 1079-1081, logical area 82     |
|                             0052  03F9  pages 1082-1084, logical area 82     |
|                             0052  03FB  pages 1085-1087, logical area 82     |
|                             0052  03FD  pages 1088-1090, logical area 82     |
|                               00  03FF  MBZ free '.'                         |
+------------------------------------------------------------------------------+
```

6.11 Transaction Recovery Duration Estimate Screen

One of the most difficult database attributes to determine is how long the database will be frozen if a process terminates prematurely, or how long a transaction rollback will take. Transaction recovery is affected by many factors, most of which are difficult to determine from run-time information available from the RMU Show Statistics command.

Therefore, the RMU Show Statistics command has been enhanced to provide an estimate of the time it will take to roll back a transaction or to completely recover a failed process.

Note: The information provided on the Transaction Recovery Duration Estimate screen is an estimate based on previous process recovery operations and other factors such as page contention and disk throughput.

This information is an estimate only; the actual process recovery duration may be more or less than described on this screen.

Individual process failure recovery performance can vary widely depending on many factors that cannot be accounted for in the displayed estimate. These factors include lock deadlock stalls, network delays, disk contention, and many other system factors, such as lock remastering.

The following example provides a sample transaction recovery scenario to consider:

```
Node: ALPH   (1/1/2)    Oracle Rdb V7.1 Perf. Monitor 17-AUG-1997 08:50:48.41
Rate: 1.00 Second    Transaction Recovery Duration Estimate Elapsed: 00:29:04.34
Page: 1 of 1            DISK$:[WORK]MF_PERSONNEL.RDB;1     Mode: Online
--------------------------------------------------------------------------------
Process.ID  RUJ.Sz Tx.Rollback DBR.Tx.Undo AIJ.Ckpt Pnd DBR.Tx.Redo DB.Freeze.Tm
2AA0ECC3:1     431 00:00:08.62 00:00:02.15 0:638    222 00:00:00.22 00:00:10.11




--------------------------------------------------------------------------------
Exit Help Menu >next_page <prev_page Refresh Set_rate Write Zoom !
```

The Transaction Recovery Duration Estimate screen provides the following information:

- Process.ID **-** This is the process identifier of a process that has the potential to roll back a transaction or require transaction recovery in the event of process failure.
- RUJ.Sz **-** This is the number of blocks of RUJ information that have been written by the process.
- Tx.Rollback **-** This is the estimate of the time it would require for the process to roll back the transaction. Note that this is different from the time it would take the DBR process to roll back the transaction.
- DBR.Tx.Undo **-** This is the estimate of the time it would require for the DBR process to undo the transaction. The DBR transaction undo duration is typically less than it takes the process to roll back the transaction, due to various optimizations and simplifications in the DBR recovery algorithm.
- AIJ.Ckpt **-** If the fast commit feature is enabled, this is the most recent checkpoint location in the AIJ journal for the process.
- Pnd **-** If AIJ journaling is enabled, this is the number of blocks of AIJ information that has been submitted (pending) but not yet written to the AIJ journal.
- DBR.Tx.Redo **-** If the fast commit feature is enabled, this is the estimate of the time it would take the DBR process to redo the failed process' previously committed transactions to the database.
- DB.Freeze.Tm **-** This is the estimate of the total time the database would be frozen if the current process were to terminate prematurely.

In the preceding example, there are three estimates of essential information:

- Process transaction rollback duration
- DBR transaction undo and redo duration
- Total database freeze duration

In the preceding example, if the process were to roll back the current transaction, it would take approximately 8 seconds. If the process were to fail prematurely, it would take the DBR process approximately 2 seconds to undo the transaction, but approximately .25 seconds to redo all previously committed transactions for that process. The total database freeze time is estimated to be approximately 10 seconds.

To validate the screen information, examine the end of the DBR log file, which is enabled using the RDM$BIND_DBR_LOG_FILE logical. For example:

```
18-AUG-1997 11:16:31.22 - TSN 0:291 was rolled back
18-AUG-1997 11:16:31.26 - Total recovery duration 10.10 seconds
```

To examine the past history of recovery operations, use the RMU Dump command with the Header qualifier and review the Database Recovery section. For example:

```
    Database Recovery...
      - 2 process failures have occurred (last 18-AUG-1997 11:16:31.26)
      - DBR freeze averaging 5.470 seconds per recovery
        Transaction REDO averaging 0.890 seconds per recovery
        Transaction UNDO averaging 3.465 seconds per recovery
        AIJ recovery averaging 1.10 seconds per recovery
        Global buffer recovery averaging 0.0 seconds per recovery
        Global buffer tx recovery averaging 0.0 seconds per recovery
        Record cache recovery averaging 0.0 seconds per recovery
      - DBR redo averaging 318 AIJ blocks per recovery
      - DBR redo recovery rate averaging 2ms per AIJ block
      - DBR undo averaging 635 RUJ blocks per recovery
      - DBR undo recovery rate averaging 5ms per RUJ block
      - DBR AIJ scan averaging 63 AIJ blocks per recovery
      - DBR AIJ scan rate averaging 1ms per AIJ block
      - Database is consistent but has been modified
      - Full AIJ roll-forward is no longer permitted to this database
  By-Area and By-Page AIJ roll-forward is permitted
      - Full AIJ roll-forward to a newly restored database is permitted
      - Next AIJ sequence number expected is 1
      - Last commit transaction TSN is 0:320
      - AIJ roll-forward is no-quiet-point enabled
```

The Transaction Recovery Duration Estimate screen is only available during online statistics collection. It is not available during binary file replay.

The configuration variable RECOVERY_SORT can be used to sort the Transaction Recovery Duration Estimate screen by specifying one of the following keywords:

- LONGEST_TRANSACTION - Sort by longest transaction rollback duration.
- LONGEST_UNDO - Sort by longest DBR undo duration estimate.
- LONGEST_REDO - Sort by longest DBR redo duration estimate.
- LONGEST_FREEZE - Sort by longest database freeze duration estimate.

These sort criteria can also be selected online using the Config on-screen menu option.

6.12 File Overview Sorting and Filtering Enhancements

The RMU Show Statistics command File IO Overview and File Lock Overview screens have been enhanced to provide additional sorting and filtering capabilities.

Two new sort options have been added to the screen configuration options obtained using the Config on-screen menu. The new Sort Alphabetically option sorts the storage area names without regard to storage area type (data or snapshot). The new Sort Alphabetically by Type option sorts the storage area names within storage area type (data or snapshot).

For example, the following File IO Overview screen shows the standard unsorted display:

```
Node: ALPH   (1/1/2)    Oracle Rdb V7.1 Perf. Monitor 25-AUG-1997 09:20:34.19
Rate: 1.00 Second    File IO Overview (Unsorted total I/O)  Elapsed: 00:04:07.54
Page: 1 of 1             DISK$:[WORK]MF_PERSONNEL.RDB;1     Mode: Online
--------------------------------------------------------------------------------
File/Storage.Area.Name........ Sync.Reads SyncWrites AsyncReads AsyncWrits PgCkd
Database Root                          17          0          0          1     0
AIJ (After-Image Journal)               0          0          0          0     0
RUJ (Recovery-Unit Journal)             0          0          0          0     0
ACE (AIJ Cache Electronic)              0          0          0          0     0
All data/snap files                     3          0          0          0     0
data JOBS                               0          0          0          0     0
data MF_PERS_DEFAULT                    3          0          0          0     0
data SALARY_HISTORY                     0          0          0          0     0
data DEPARTMENTS                        0          0          0          0     0
data EMPIDS_LOW                         0          0          0          0     0
data EMPIDS_MID                         0          0          0          0     0
data EMPIDS_OVER                        0          0          0          0     0
data EMP_INFO                           0          0          0          0     0
```

```
data MF_PERS_SEGSTR                0          0          0          0     0
snap JOBS                          0          0          0          0     0
snap MF_PERS_DEFAULT               0          0          0          0     0
snap SALARY_HISTORY                0          0          0          0     0
snap DEPARTMENTS                   0          0          0          0     0
snap EMPIDS_LOW                    0          0          0          0     0
snap EMPIDS_MID                    0          0          0          0     0
snap EMPIDS_OVER                   0          0          0          0     0
snap EMP_INFO                      0          0          0          0     0
snap MF_PERS_SEGSTR                0          0          0          0     0
--------------------------------------------------------------------------------
Config Exit Filter Help Menu >next_page <prev_page Options Reset Set_rate Write
```

The following File IO Overview screen shows the display sorted alphabetically:

```
Node: ALPH   (1/1/2)    Oracle Rdb V7.1 Perf. Monitor 25-AUG-1997 09:20:40.92
Rate: 1.00 Second       File IO Overview (Alphabetical)     Elapsed: 00:04:14.27
Page: 1 of 1             DISK$:[WORK]MF_PERSONNEL.RDB;1     Mode: Online
--------------------------------------------------------------------------------
File/Storage.Area.Name........ Sync.Reads SyncWrites AsyncReads AsyncWrits PgCkd
ACE (AIJ Cache Electronic)             0          0          0          0     0
AIJ (After-Image Journal)              0          0          0          0     0
All data/snap files                    3          0          0          0     0
data DEPARTMENTS                       0          0          0          0     0
snap DEPARTMENTS                       0          0          0          0     0
Database Root                         17          0          0          1     0
data EMPIDS_LOW                        0          0          0          0     0
snap EMPIDS_LOW                        0          0          0          0     0
data EMPIDS_MID                        0          0          0          0     0
snap EMPIDS_MID                        0          0          0          0     0
data EMPIDS_OVER                       0          0          0          0     0
snap EMPIDS_OVER                       0          0          0          0     0
data EMP_INFO                          0          0          0          0     0
snap EMP_INFO                          0          0          0          0     0
data JOBS                              0          0          0          0     0
snap JOBS                              0          0          0          0     0
data MF_PERS_DEFAULT                   3          0          0          0     0
snap MF_PERS_DEFAULT                   0          0          0          0     0
data MF_PERS_SEGSTR                    0          0          0          0     0
snap MF_PERS_SEGSTR                    0          0          0          0     0
RUJ (Recovery-Unit Journal)            0          0          0          0     0
data SALARY_HISTORY                    0          0          0          0     0
snap SALARY_HISTORY                    0          0          0          0     0
 ----------------------------------------------------------------------------
Config Exit Filter Help Menu >next_page <prev_page Options Reset Set_rate Write
```

The following File IO Overview screen shows the display sorted alphabetically by type:

```
Node: ALPH   (1/1/2)    Oracle Rdb V7.1 Perf. Monitor 25-AUG-1997 09:20:44.42
Rate: 1.00 Second       File IO Overview (Alphabetical)     Elapsed: 00:04:17.77
Page: 1 of 1             DISK$:[WORK]MF_PERSONNEL.RDB;1     Mode: Online
--------------------------------------------------------------------------------
File/Storage.Area.Name........ Sync.Reads SyncWrites AsyncReads AsyncWrits PgCkd
ACE (AIJ Cache Electronic)              0          0          0          0     0
AIJ (After-Image Journal)               0          0          0          0     0
All data/snap files                     3          0          0          0     0
Database Root                          17          0          0          1     0
RUJ (Recovery-Unit Journal)             0          0          0          0     0
data DEPARTMENTS                        0          0          0          0     0
data EMPIDS_LOW                         0          0          0          0     0
data EMPIDS_MID                         0          0          0          0     0
data EMPIDS_OVER                        0          0          0          0     0
data EMP_INFO                           0          0          0          0     0
data JOBS                               0          0          0          0     0
data MF_PERS_DEFAULT                    3          0          0          0     0
data MF_PERS_SEGSTR                     0          0          0          0     0
data SALARY_HISTORY                     0          0          0          0     0
snap DEPARTMENTS                        0          0          0          0     0
snap EMPIDS_LOW                         0          0          0          0     0
snap EMPIDS_MID                         0          0          0          0     0
snap EMPIDS_OVER                        0          0          0          0     0
snap EMP_INFO                           0          0          0          0     0
snap JOBS                               0          0          0          0     0
snap MF_PERS_DEFAULT                    0          0          0          0     0
snap MF_PERS_SEGSTR                     0          0          0          0     0
snap SALARY_HISTORY                     0          0          0          0     0
--------------------------------------------------------------------------------
Config Exit Filter Help Menu >next_page <prev_page Options Reset Set_rate Write
```

Also, a new Filter on-screen menu option has been added. The Filter on-screen menu option prompts the user to enter a pattern string that includes wildcard characters. By using wildcard characters in the search pattern, for example, it is possible to find all EMP storage areas using the search pattern "*EMP*".

Note: Search patterns specified without wildcard characters find exact matches only. For example, the wildcard name "EMP" finds the single storage area whose name is "EMP."

The pattern string can contain either one or both of the two wildcard characters, asterisk (*) and percent sign (%). The asterisk is mapped to zero or more characters. The percent sign is mapped to only one character.

You can enter a different filter for each screen.

Filtering of the alphabetically sorted storage areas is permitted.

When you specify a filter, the Filter on-screen menu option is highlighted. Select the Filter on-screen menu option and press the Return key to delete any previously existing filter.

The following example shows the File IO Overview screen filtered using the pattern "*EMP*":

```
Node: ALPH   (1/1/2)    Oracle Rdb V7.1 Perf. Monitor 25-AUG-1997 09:25:50.61
Rate: 1.00 Second    File IO Overview (Unsorted total I/O)  Elapsed: 00:00:05.57
Page: 1 of 1             DISK$:[WORK]MF_PERSONNEL.RDB;1     Mode: Online
--------------------------------------------------------------------------------
File/Storage.Area.Name........ Sync.Reads SyncWrites AsyncReads AsyncWrits PgCkd
data EMPIDS_LOW                         0          0          0          0     0
data EMPIDS_MID                         0          0          0          0     0
data EMPIDS_OVER                        0          0          0          0     0
data EMP_INFO                           0          0          0          0     0
snap EMPIDS_LOW                         0          0          0          0     0
snap EMPIDS_MID                         0          0          0          0     0
snap EMPIDS_OVER                        0          0          0          0     0
snap EMP_INFO                           0          0          0          0     0


--------------------------------------------------------------------------------
Config Exit Filter Help Menu >next_page <prev_page Options Reset Set_rate Write
```

Note: The data and snap prefixes are not part of the storage area name and are not considered when applying a specified filter. For example, the pattern "data*" does not find all data storage areas.

To control the selection of storage area types, three new sort options have been added to the screen configuration options obtained using the Config on-screen menu. The new Display All Storage Areas option displays all storage areas. The new Display Data Storage Areas Only option displays only live data storage areas. The new Display Snap Storage Areas Only option displays only snapshot storage areas.

The following example shows the File IO Overview screen displaying only live storage areas:

```
Node: ALPH   (1/1/2)    Oracle Rdb V7.1 Perf. Monitor 25-AUG-1997 13:46:22.48
Rate: 1.00 Second    File IO Overview (Unsorted total I/O)  Elapsed: 00:01:46.60
Page: 1 of 1             DISK$:[WORK]MF_PERSONNEL.RDB;1     Mode: Online
--------------------------------------------------------------------------------
File/Storage.Area.Name........ Sync.Reads SyncWrites AsyncReads AsyncWrits PgCkd
data MF_PERS_DEFAULT                    3          0          0          0     0
data DEPARTMENTS                        0          0          0          0     0
data EMPIDS_LOW                         0          0          0          0     0
data EMPIDS_MID                         0          0          0          0     0
data EMPIDS_OVER                        0          0          0          0     0
data EMP_INFO                           0          0          0          0     0
data JOBS                               0          0          0          0     0
data MF_PERS_SEGSTR                     0          0          0          0     0
data SALARY_HISTORY                     0          0          0          0     0
--------------------------------------------------------------------------------
Config Exit Filter Help Menu >next_page <prev_page Options Reset Set_rate Write
```

The following example shows the File IO Overview screen displaying only snapshot storage areas:

```
Node: ALPH   (1/1/2)    Oracle Rdb V7.1 Perf. Monitor 25-AUG-1997 13:46:26.06
Rate: 1.00 Second    File IO Overview (Unsorted total I/O)  Elapsed: 00:01:50.18
Page: 1 of 1             DISK$:[WORK]MF_PERSONNEL.RDB;1     Mode: Online
--------------------------------------------------------------------------------
File/Storage.Area.Name........ Sync.Reads SyncWrites AsyncReads AsyncWrits PgCkd
snap MF_PERS_DEFAULT                    0          0          0          0     0
snap DEPARTMENTS                        0          0          0          0     0
snap EMPIDS_LOW                         0          0          0          0     0
snap EMPIDS_MID                         0          0          0          0     0
snap EMPIDS_OVER                        0          0          0          0     0
snap EMP_INFO                           0          0          0          0     0
snap JOBS                               0          0          0          0     0
snap MF_PERS_SEGSTR                     0          0          0          0     0
snap SALARY_HISTORY                     0          0          0          0     0
--------------------------------------------------------------------------------
Config Exit Filter Help Menu >next_page <prev_page Options Reset Set_rate Write
```

6.13 Fast Incremental Backup Display

The RMU Show Statistics command has been enhanced to display fast incremental backup run-time statistics in the Fast Incr Backup Statistics screen, located in the Journaling Information submenu.

The following is an example of the Fast Incr Backup Statistics screen:

```
Node: ALPH   (1/1/2)    Oracle Rdb V7.1 Perf. Monitor 11-SEP-1997 13:45:05.69
Rate: 0.50 Seconds        Fast Incr Backup Statistics       Elapsed: 00:35:38.17
Page: 1 of 1            DISK$:[WORK]MF_PERSONNEL.RDB;1      Mode: Online
--------------------------------------------------------------------------------

statistic.........      rate.per.second............. total....... average......
name..............      max..... cur..... avg....... count....... per.trans....

FIB update attempt            32        0       10.3       22033           1.6
FIB map updated                0        0        0.0          15           0.0
SPAM page updated              0        0        0.0          15           0.0
SPAM updt deferred            32        0       10.2       22015           1.6
--------------------------------------------------------------------------------
Exit Graph Help Menu Options Pause Reset Set_rate Time_plot Write X_plot Yank !
```

The following explains the statistical information displayed:

- FIB update attempt - Indicates the number of times the fast incremental backup (FIB) update operation was attempted. The attempt does not always result in the SPAM page being updated.
- FIB map updated - Indicates the number of times the FIB map, a per-process data structure, was updated. This data structure indicates when each process no longer needs to update a particular SPAM page.
- SPAM page updated - Indicates the number of times a SPAM page was immediately modified to indicate that one or more pages in the SPAM interval have been modified since the last incremental backup. Each SPAM page update results in one synchronous read I/O and one synchronous write I/O operation.
- SPAM updt deferred - Indicates the number of times a SPAM page did not need to be immediately modified, but might have to be modified at a later time. In most cases, this statistic closely follows the FIB update attempt statistic.

This screen is available during replay of a binary input file and is also available clusterwide.

6.14 Hot Standby Database Recovery Information

The RMU Show Statistics command has been enhanced to display statistics about the recovery of a hot standby database. You can select this Recovery Information screen from the Hot Standby Information menu when hot standby is active. This screen is available on the standby database only.

The Recovery Information screen is important for analyzing network throughput and the effectiveness of hot standby database resource allocation.

The Recovery Information screen is available during replay of a binary input file and is integrated into the Cluster Statistic Collection facility. This screen does not have a zoom screen.

The following is an example of the Recovery Information screen:

```
Node: BONZAI (1/1/24)   Oracle Rdb V7.1    Perf. Monitor  8-JUN-1998 11:08:23.96
Rate: 1.00 Second            Recovery Information          Elapsed: 05:08:48.28
Page: 1 of 1          USER1:[SPANDERSON.TCS_STANDBY]TCS.RDB;1       Mode: Online
--------------------------------------------------------------------------------
statistic.........      rate.per.second............. total....... average......
name..............     max..... cur..... avg....... count....... per.trans....

transactions                22        0        0.4        7571          1.0
 commit                     22        0        0.4        7571          1.0
 rollback                    0        0        0.0           0          0.0
 prepared                    0        0        0.0           0          0.0

Area ready                   0        0        0.0           7          0.0

AIJ records                781        0        3.9       73127          9.6
 erase mixed                 0        0        0.0           0          0.0
 erase uniform               2        0        0.0          39          0.0
 modify mixed                0        0        0.0           0          0.0
 modify uniform            781        0        3.9       73088          9.6

SPAM updated                93        0        0.7       13017          1.7
--------------------------------------------------------------------------------
Exit Graph Help Menu Options Pause Reset Set_rate Time_plot Write X_plot Yank !
```

The following list describes the statistical information displayed:

- transactions **-** Displays the number of completed database transactions. (The count of the COMMIT and ROLLBACK statements that have executed.)

- commit - Displays the number of transactions that have been committed to the standby database.
- rollback - Displays the number of transactions that have been rolled back prior to being applied to the standby database.
- prepared - Displays the number of distributed transactions that have been successfully prepared in anticipation of eventually being committed to the standby database.
- Area ready - Displays the number of physical storage areas that have been readied during the recovery operation.
- AIJ records - Displays the number of AIJ records applied.
- erase mixed - Displays the number of erase record operations performed on a mixed-format storage area.
- erase uniform - Displays the number of erase record operations performed on a uniform-format storage area.
- modify mixed - Displays the number of modify record operations performed on a mixed-format storage area.
- modify uniform - Displays the number of modify record operations performed on a uniform-format storage area.
- SPAM updated - Displays the number of SPAM page modifications that occurred as a result of the AIJ journal record. SPAM pages are typically modified when a live data page changes its threshold information.

6.15 Transaction Activity and Execution Rate Information

The RMU Show Statistics command has been enhanced to display statistics that summarize database transaction activity and transaction and verb execution rates. You can select this Summary Tx Statistics screen from the Main menu.

The Summary Tx Statistics screen is available during replay of a binary input file and is integrated into the Cluster Statistics Collection facility. This screen does not have a zoom screen.

When viewing this screen, compare the verb successes statistic to the verb failures statistic. A high failure rate compared to the success rate indicates that many things are failing. The problem might be too many deadlocks, but this is application-dependent.

Compare the RUJ file reads and the RUJ file writes statistics. Too many RUJ file reads indicate too many rollbacks. Too many RUJ file writes might indicate data page contention problems or transactions that modify a lot of data. More than one write for each transaction indicates that marked pages are being written back to the database before the actual commit operation. This means that another recovery-unit requested the page, the buffer pool overflowed, or there are very large transactions with a lot of updates.

Examine the rolled back duration x100 field to ensure that transaction rollback operations are not excessively long. (To identify transactions with potentially long transaction rollback durations, carefully review the Transaction Recovery Duration Estimate screen, located in the Process Information submenu.)

The following is an example of the Summary Tx Statistics screen:

```
Node: ALPHA3 (1/1/24)   Oracle Rdb V7.1    Perf. Monitor  8-JUN-1998 08:41:35.40
Rate: 1.00 Second           Summary Tx Statistics         Elapsed: 02:40:31.18
Page: 1 of 1          USER1:[SPANDERSON.TCS_MASTER]TCS.RDB;1       Mode: Online
--------------------------------------------------------------------------------
statistic.........      rate.per.second............. total....... average......
name..............      max..... cur..... avg....... count....... per.trans....
transactions                   0        0        0.0            3           1.0
  committed                    0        0        0.0            3           1.0
  rolled back                  0        0        0.0            0           0.0
    duration x100              0        0        0.0            0           0.0
  prepared                     0        0        0.0            0           0.0

verb successes                 0        0        0.0            9           3.0
verb failures                  0        0        0.0            0           0.0
  duration x100                0        0        0.0            0           0.0

checkpoints                    0        0        0.0            0           0.0
   duration x100               0        0        0.0            0           0.0

RUJ file reads                 0        0        0.0            0           0.0
    file writes                0        0        0.0            3           1.0
    file extend                0        0        0.0            0           0.0
--------------------------------------------------------------------------------
Exit Graph Help Menu Options Pause Reset Set_rate Time_plot Write X_plot Yank !
```

The following list describes the statistical information displayed:

- transactions - Displays the number of completed database transactions. This is the count of the COMMIT and ROLLBACK statements that have executed.

- committed - Identifies the actual number of transactions that committed successfully to the database.
- rolled back - Identifies the actual number of transactions that aborted and were not applied to the database.
- duration x100 - Identifies the duration of a transaction rollback operation, verb failure rollback operation, or checkpoint operation expressed in hundredths of a second and displayed as a whole number. For example, the value 500 equates to 5 seconds.
- prepared - Identifies the number of distributed transactions that have successfully prepared themselves for subsequent transaction commit.
- verb successes - Displays the number of completed verbs that returned a successful status code.

  A **verb** is an atomic SQL statement or action. For example, a row insert is a verb. Within a compound statement, each individual statement is atomic and Oracle Rdb performs a verb success operation after processing each one. To avoid this overhead, use the SQL BEGIN atomic statement to treat the entire block as a single verb.
- verb failures - Displays the number of completed verbs that returned an error status code. Errors include end-of-collection and deadlocks, as well as all other exception conditions.

  Excessive verb failures are usually an indication of a failed constraint, such as uniqueness criteria, or an invalid DDL statement. Note that in the case of cursors and scans, reaching the end-of-stream always results in a verb failure.

  Note that SQL performs its own internal queries to identify metadata, such as relation or index names.

  Oracle Rdb rarely issues a verb failure unless there is an exception of some kind, such as a constraint failure.
- checkpoints - Identifies the number of checkpoints performed by users. This field does not include the initial checkpoint that occurs when a user first attaches to the database.
- RUJ file reads - Displays the total number of read I/O operations performed on the RUJ journal during the transaction undo phase. The .ruj file is never written by the Database Recovery process (DBR).

  This field includes both synchronous and asynchronous I/O read requests.

- file writes - Displays the total number of write I/O operations performed on the RUJ journal during the transaction phase.

  This field includes both synchronous and asynchronous I/O write requests.
- file extend - Identifies the number of times an .ruj file has been extended.

6.16 Detail Added to PIO Statistics - SPAM Access Screen

The RMU Show Statistics command has been enhanced with the PIO Statistics--SPAM Access screen. The purpose of this screen is to identify the reason why the SPAM was accessed (for either read or write access).

Consider the following screen:

```
Node: ALPH   (1/1/16)   Oracle Rdb V7.1 Perf. Monitor  5-JAN-1998 11:29:21.13
Rate: 1.00 Second         PIO Statistics--SPAM Access       Elapsed: 00:31:47.55
Page: 1 of 1            DISK$:[WORK]MF_PERSONNEL.RDB;1      Mode: Online
--------------------------------------------------------------------------------

statistic.........      rate.per.second............. total....... average......
name..............      max..... cur..... avg....... count....... per.trans....
fetch for read                20        0        9.3       17821            1.0
 uniform area scan            15        0        0.1         280            0.0
 record store fet             20        0        9.1       17541            1.0
 record modify fet             0        0        0.0           0            0.0
 record erase fet              0        0        0.0           0            0.0
fetch for write               47        0        1.1        2250            0.1
 record store upd              4        0        0.4         858            0.0
 record modify upd             0        0        0.0           0            0.0
 record erase upd             23        0        0.1         321            0.0
fetch for update              47        0        1.1        2250            0.1
 clump allocate                3        0        0.1         216            0.0
 fast incr. bkup               0        0        0.0           0            0.0
 threshold update             23        0        0.5         963            0.0
record stored                 16        0        8.7       16677            1.0
record marked               1849        0       22.0       42049            2.5
record erased                622        0        4.3        8338            0.5
--------------------------------------------------------------------------------
Exit Graph Help Menu Options Pause Reset Set_rate Time_plot Write X_plot Yank !
```

As can be seen in the preceding example, the majority of the SPAM page "fetch for read" accesses were caused by record storage. The SPAM page "fetch for write" accesses are evenly distributed between record stores and record erases.

Note that 16,677 records were stored, while 17,541 SPAM fetches occurred because of those stores. However, only 858 of those SPAM fetches actually resulted in updates to the SPAM thresholds.

You can identify excessive SPAM fetches by comparing the "record store fet" field to the "record store upd" and "record stored" fields. The following list describes the SPAM access screen fields.

- fetch for read - The total number of times the SPAM page was fetched for retrieval.
- uniform area scan - The total number of times the SPAM page was fetched for retrieval during a uniform area scan. This is used primarily to check if SPAM thresholds need to be adjusted.
- record store fet - The total number of times the SPAM page was fetched for retrieval during a record store operation. This is primarily used to check if SPAM thresholds need to be adjusted.
- record modify fet - The total number of times the SPAM page was fetched for retrieval during a record modification operation. This is primarily used to check if SPAM thresholds need to be adjusted.
- record erase fet - The total number of times the SPAM page was fetched for retrieval during a record erase operation. This is primarily used to check if SPAM thresholds need to be adjusted.
- fetch for write - The total number of times the SPAM page was fetched for update.
- record store upd - The total number of times the SPAM page was fetched for update during a record store operation. This is primarily used to modify the SPAM thresholds.
- record modify upd - The total number of times the SPAM page was fetched for update during a record modification operation. This is primarily used to modify the SPAM thresholds.
- record erase upd - The total number of times the SPAM page was fetched for update during a record erase operation. This is primarily used to modify the SPAM thresholds.
- fetch for update - The total number of times the SPAM page was fetched for update.
- clump allocate - The total number of times the SPAM page was updated for a clump allocation operation.

- fast incr. bkup - The total number of times the SPAM page was updated for a fast incremental backup modification.
- threshold update - The total number of times the SPAM page was updated to change threshold information for a data page.
- record stored - The total number of records stored.
- record marked - The total number of records modified.
- record erased - The total number of records erased.

The PIO Statistics--SPAM Access screen is recorded to the binary output file and is available during binary input file replay.

The following example shows the statistics collected following an operation that stored 8,192 records into a uniform-format storage area:

```
Node: ALPH   (1/1/16)   Oracle Rdb V7.1 Perf. Monitor  5-JAN-1998 12:20:06.57
Rate: 1.00 Second         PIO Statistics--SPAM Access       Elapsed: 00:10:42.88
Page: 1 of 1            DISK$:[WORK]MF_PERSONNEL.RDB;1      Mode: Online
--------------------------------------------------------------------------------
statistic.........      rate.per.second............. total....... average......
name..............      max..... cur..... avg....... count....... per.trans....
fetch for read                19        0       13.6         8775            1.0
 uniform area scan             1        0        0.1          110            0.0
 record store fet             17        0       13.4         8665            1.0
 record modify fet             0        0        0.0            0            0.0
 record erase fet              0        0        0.0            0            0.0
fetch for write                3        0        0.9          642            0.0
 record store upd              1        0        0.4          321            0.0
 record modify upd             0        0        0.0            0            0.0
 record erase upd              0        0        0.0            0            0.0
fetch for update               3        0        0.9          642            0.0
 clump allocate                0        0        0.0            0            0.0
 fast incr. bkup               0        0        0.0            0            0.0
 threshold update              1        0        0.4          321            0.0
record stored                 16        0       12.9         8192            1.0
record marked                 17        0       13.4         8661            1.0
record erased                  0        0        0.0            0            0.0
--------------------------------------------------------------------------------
Exit Graph Help Menu Options Pause Reset Set_rate Time_plot Unreset Write X_plot
```

The following example shows the statistics collected following an operation that scanned 8,192 records into an uniform-format storage area:

```
Node: ALPH   (1/1/16)   Oracle Rdb V7.1 Perf. Monitor  5-JAN-1998 12:42:44.65
```

```
Rate: 1.00 Second        PIO Statistics--SPAM Access      Elapsed: 00:01:25.56
Page: 1 of 1            DISK$:[WORK]MF_PERSONNEL.RDB;1      Mode: Online
--------------------------------------------------------------------------------

statistic.........      rate.per.second............. total....... average......
name..............      max..... cur..... avg....... count....... per.trans....
fetch for read              158        0        3.8          330         165.0
 uniform area scan          158        0        3.8          330         165.0
 record store fet             0        0        0.0            0           0.0
 record modify fet            0        0        0.0            0           0.0
 record erase fet             0        0        0.0            0           0.0
fetch for write               0        0        0.0            0           0.0
 record store upd             0        0        0.0            0           0.0
 record modify upd            0        0        0.0            0           0.0
 record erase upd             0        0        0.0            0           0.0
fetch for update              0        0        0.0            0           0.0
 clump allocate               0        0        0.0            0           0.0
 fast incr. bkup              0        0        0.0            0           0.0
 threshold update             0        0        0.0            0           0.0
record stored                 0        0        0.0            0           0.0
record marked                 0        0        0.0            0           0.0
record erased                 0        0        0.0            0           0.0
--------------------------------------------------------------------------------
Exit Graph Help Menu Options Pause Reset Set_rate Time_plot Unreset Write X_plot
```

The following example shows the statistics collected following an operation that modified 8,192 records into an uniform-format storage area:

```
Node: ALPH   (1/1/16)   Oracle Rdb V7.1 Perf. Monitor  5-JAN-1998 12:24:44.15
Rate: 1.00 Second        PIO Statistics--SPAM Access      Elapsed: 00:03:34.91
Page: 1 of 1            DISK$:[WORK]MF_PERSONNEL.RDB;1      Mode: Online
--------------------------------------------------------------------------------

statistic.........      rate.per.second............. total....... average......
name..............      max..... cur..... avg....... count....... per.trans....
fetch for read              543        0       31.4         6765        3382.5
 uniform area scan          147        0        1.9          415         207.5
 record store fet           529        0       29.5         6350        3175.0
 record modify fet            0        0        0.0            0           0.0
 record erase fet             0        0        0.0            0           0.0
fetch for write              63        0        3.3          711         355.5
 record store upd            34        0        1.8          395         197.5
 record modify upd            0        0        0.0            0           0.0
 record erase upd             0        0        0.0            0           0.0
fetch for update             63        0        3.3          711         355.5
```

```
 clump allocate             14        0       0.7          158         79.0
 fast incr. bkup             0        0       0.0            0          0.0
 threshold update           20        0       1.1          237        118.5
record stored              494        0      26.5         5703       2851.5
record marked              703        0      38.1         8192       4096.0
record erased                0        0       0.0            0          0.0

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

The following example shows the number of SPAM pages retrieved in order to store a single, new record:

```
Node: ALPH   (1/1/16)   Oracle Rdb V7.1 Perf. Monitor  5-JAN-1998 12:25:55.48
Rate: 1.00 Second         PIO Statistics--SPAM Access       Elapsed: 00:00:24.01
Page: 1 of 1            DISK$:[WORK]MF_PERSONNEL.RDB;1      Mode: Online
--------------------------------------------------------------------------------

statistic.........      rate.per.second............. total....... average......
name..............      max..... cur..... avg....... count....... per.trans....
fetch for read             720        0      31.4          756        756.0
 uniform area scan         180        0       7.9          190        190.0
 record store fet          539        0      23.5          566        566.0
 record modify fet           0        0       0.0            0          0.0
 record erase fet            0        0       0.0            0          0.0
fetch for write              1        0       0.0            2          2.0
 record store upd            0        0       0.0            1          1.0
 record modify upd           0        0       0.0            0          0.0
 record erase upd            0        0       0.0            0          0.0
fetch for update             1        0       0.0            2          2.0
 clump allocate              0        0       0.0            0          0.0
 fast incr. bkup             0        0       0.0            0          0.0
 threshold update            0        0       0.0            1          1.0
record stored                1        0       0.0            2          2.0
record marked                1        0       0.0            2          2.0
record erased                0        0       0.0            0          0.0

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

After a clump has been allocated, subsequent record storage into the same clump is significantly easier, as shown in the following example:

```
Node: ALPH   (1/1/16)   Oracle Rdb V7.1 Perf. Monitor  5-JAN-1998 12:27:05.47
Rate: 1.00 Second         PIO Statistics--SPAM Access       Elapsed: 00:00:36.53
```

```
Page: 1 of 1           DISK$:[WORK]MF_PERSONNEL.RDB;1      Mode: Online
--------------------------------------------------------------------------------

statistic.........     rate.per.second............. total....... average......
name..............     max..... cur..... avg....... count....... per.trans....
fetch for read              621        0      17.8          653         326.5
 uniform area scan          156        0       4.4          164          82.0
 record store fet           465        0      13.3          489         244.5
 record modify fet            0        0       0.0            0           0.0
 record erase fet             0        0       0.0            0           0.0
fetch for write               0        0       0.0            0           0.0
 record store upd             0        0       0.0            0           0.0
 record modify upd            0        0       0.0            0           0.0
 record erase upd             0        0       0.0            0           0.0
fetch for update              0        0       0.0            0           0.0
 clump allocate               0        0       0.0            0           0.0
 fast incr. bkup              0        0       0.0            0           0.0
 threshold update             0        0       0.0            0           0.0
record stored                 0        0       0.0            1           0.5
record marked                 0        0       0.0            1           0.5
record erased                 0        0       0.0            0           0.0

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

Index

B

C

D

E

F

G

H

I

M

N

O

P

Q

R

S

T

U

V

W

X

Z