Oracle® Rdb7 for OpenVMS

Release Notes

Release 7.0.6.1

February 2001

ORACLE®

Oracle Rdb7 Release Notes, Release 7.0.6.1 for OpenVMS

This document was prepared using VAX DOCUMENT Version 2.1.

Contents

3 Enhancements

4 Documentation Corrections

5 Known Problems and Restrictions

Examples

Tables

Preface

Purpose of This Manual

This manual contains release notes for Oracle Rdb7 Release 7.0.6.1. The notes describe changed and enhanced features; upgrade and compatibility information; new and existing software problems and restrictions; and software and documentation corrections. These release notes cover both Oracle Rdb7 for OpenVMS Alpha and Oracle Rdb7 for OpenVMS VAX, which are referred to by their abbreviated name, Oracle Rdb7.

Intended Audience

This manual is intended for use by all Oracle Rdb7 users. Read this manual before you install, upgrade, or use Oracle Rdb7 Release 7.0.6.1.

Document Structure

This manual consists of five chapters:

Chapter 1	Describes how to install Oracle Rdb7 Release 7.0.6.1.
Chapter 2	Describes software errors corrected in Oracle Rdb7 Release 7.0.6.1
Chapter 3	Describes enhancements introduced in Oracle Rdb7 Release 7.0.6.1.
Chapter 4	Provides information not currently available in the Oracle Rdb7 documentation set.
Chapter 5	Describes problems, restrictions, and workarounds known to exist in Oracle Rdb7 Release 7.0.6.1.

1

Installing Oracle Rdb7 Release 7.0.6.1

This software update is installed using the standard OpenVMS Install Utility.

1.1 Requirements

The following conditions must be met in order to install this software update:

- Oracle Rdb7 must be shutdown before you install this update kit. That is, the command file SYS$STARTUP:RMONSTOP(70).COM should be executed before proceeding with this installation. If you have an OpenVMS cluster, you must shutdown all versions of Oracle Rdb7 on all nodes in the cluster before proceeding.
- Oracle Rdb7 Release 7.0.6 must have already been installed on your system.
- The installation requires approximately 100,000 free blocks on your system disk for OpenVMS VAX systems; 200,000 blocks for OpenVMS Alpha systems.

1.2 Invoking VMSINSTAL

To start the installation procedure, invoke the VMSINSTAL command procedure:

```
@SYS$UPDATE:VMSINSTAL variant-name device-name OPTIONS N
```

variant-name

The variant names for the software update for Oracle Rdb7 Release 7.0.6.1 are:

- RDBSE1F070 for Oracle Rdb7 for OpenVMS VAX standard version.
- RDBASE1F070 for Oracle Rdb7 for OpenVMS Alpha standard version.
- RDBMVE1F070 for Oracle Rdb7 for OpenVMS VAX multiversion.
- RDBAMVE1F070 for Oracle Rdb7 for OpenVMS Alpha multiversion.

device-name

Use the name of the device on which the media is mounted.

- If the device is a disk drive, such as a CD-ROM reader, you also need to specify a directory. For CD-ROM distribution, the directory name is the same as the variant name. For example:

  ```
  DKA400:[RDBSE1F070.KIT]
  ```

- If the device is a magnetic tape drive, you need to specify only the device name. For example:

  ```
  MTA0:
  ```

OPTIONS N

This parameter prints the release notes.

The following example shows how to start the installation of the VAX standard kit on device MTA0: and print the release notes:

```
$ @SYS$UPDATE:VMSINSTAL RDBSE1F070 MTA0: OPTIONS N
```

1.3 Stopping the Installation

To stop the installation procedure at any time, press Ctrl/Y. When you press Ctrl/Y, the installation procedure deletes all files it has created up to that point and exits. You can then start the installation again.

If VMSINSTAL detects any problems during the installation, it notifies you and a prompt asks if you want to continue. You might want to continue the installation to see if any additional problems occur. However, the copy of Oracle Rdb7 installed will probably not be usable.

1.4 After Installing Oracle Rdb7

This update provides a new Oracle Rdb7 Oracle TRACE facility definition. Any Oracle TRACE selections that reference Oracle Rdb7 will need to be redefined to reflect the new facility version number for the updated Oracle Rdb7 facility definition, "RDBVMSV7.0-61".

If you have Oracle TRACE installed on your system and you would like to collect for Oracle Rdb7, you must insert the new Oracle Rdb7 facility definition included with this update kit.

The installation procedure inserts the Oracle Rdb7 facility definition into a library file called EPC$FACILITY.TLB. To be able to collect Oracle Rdb7 event-data using Oracle TRACE, you must move this facility definition into the Oracle TRACE administration database. Perform the following steps:

1. Extract the definition from the facility library to a file (in this case, RDBVMS.EPC$DEF).

   ```
   $ LIBRARY /TEXT /EXTRACT=RDBVMSV7.0-61 -
   _$ /OUT=RDBVMS.EPC$DEF SYS$SHARE:EPC$FACILITY.TLB
   ```

2. Insert the facility definition into the Oracle TRACE administration database.

   ```
   $ COLLECT INSERT DEFINITION RDBVMS.EPC$DEF /REPLACE
   ```

Note that if you are installing the multiversion variant of Oracle Rdb7, the process executing the INSERT DEFINITION command must use the version of Oracle Rdb7 that matches the version used to create the Oracle TRACE administration database or the INSERT DEFINITION command will fail.

1.5 Maximum OpenVMS Version Check Added

As of Oracle Rdb7 Release 7.0.1.5, a maximum OpenVMS version check has been added to the product. Oracle Rdb has always had a minimum OpenVMS version requirement. With 7.0.1.5 and for all future Oracle Rdb releases, we have expanded this concept to include a maximum VMS version check and a maximum supported processor hardware check. The reason for this check is to improve product quality.

OpenVMS Version 7.3 is the maximum supported version of OpenVMS.

As of Oracle Rdb7 Release 7.0.3, the Alpha EV6 processor is supported. As of Oracle Rdb7 Release 7.0.5, the Alpha EV67 processor is supported. As of Oracle Rdb7 Release 7.0.6, the Alpha Wildfire processor is supported (see http://metalink.oracle.com for specifics on which Wildfire configurations are supported).

The check for the OpenVMS operating system version and supported hardware platforms is performed both at installation time and at runtime. If either a non-certified version of OpenVMS or hardware platform is detected during installation, the installation will abort. If a non-certified version of OpenVMS or hardware platform is detected at runtime, Oracle Rdb will not start.

2

Software Errors Fixed in Oracle Rdb7 Release 7.0.6.1

This chapter describes software errors that are fixed by Oracle Rdb7 Release 7.0.6.1.

2.1 Software Errors Fixed That Apply to All Interfaces

2.1.1 Excessive Pages Checked/Discarded When Storing New Rows

Bug 1391003

Oracle Rdb7 would sometimes not use a page for storing a new row even though there was sufficient space on the page to store the new row. This would only happen when there was sufficient locked space on the page to store the data portion of the row, but insufficient free space to create the line and transaction index (LDX/TDX) entries on the page. Oracle Rdb7 would reject the page and continue checking pages until it found a page that had enough free space to store the LDX/TDX entries.

"Locked space" is space that is reserved to a specific database attach (user) and cannot be reclaimed by any other user until the user that has it locked no longer needs it; "free space" is available space that is not reserved to any particular database attach.

This problem was more likely to occur in storage areas that had been carefully sized. In that situation, it was common to have sufficient locked space on a page to store the row data and the LDX/TDX entries, but have no free space available.

This problem has been corrected in Oracle Rdb7 Release 7.0.6.1. When necessary, Oracle Rdb7 will utilize locked space when allocating LDX/TDX entries.

2.1.2 Quota Exceeded Conditions During DAPF May Lead To Missing Updates

Bug 1485622

When the Detected Asynchronous Prefetch (DAPF) feature is enabled, it is possible for certain process quota exceeded conditions to result in potential data loss. If a Detected Asynchronous Prefetch I/O request fails due to an exceeded quota error, a database page within the process buffer pool may be errantly released. This can lead to database record modifications or additions being lost.

Workarounds include disabling the Detected Asynchronous Prefetch feature. The logical name RDM$BIND_DAPF_ENABLED can be defined to a value of "0" to disable Detected Asynchronous Prefetch.

Further, accounts and processes that access Oracle Rdb databases should be reviewed to ensure that various quotas are set to ensure high levels of I/O performance. Table 2–1 lists suggested quota values for maximum performance of Rdb I/O operations.

Table 2–1 Recommended Minimum Process Quotas

Quota	Setting
DIOLM	Equal to or greater than the count of database buffers (either the database default or the setting of the logical name RDM$BIND_ BUFFERS) when local buffers are enabled for a database or a value greater than the global buffer USER LIMIT setting. Minimum of 250.
BIOLM	Equal to or greater than the setting of DIOLM.
ASTLM	Equal to or greater than 50 more than the setting of DIOLM.
BYTLM	Depending on the amount of asynchronous I/O activity, this may need to be equal to or greater than 512 times the database buffer size times one quarter the value of database buffers (either the database default or the setting of the logical name RDM$BIND_BUFFERS) when local buffers are enabled for a database or a value greater than the global buffer USER LIMIT setting. Based on a 12 block buffer size and the desire to have a process have up to 40 asynchronous I/O requests outstanding (either reading or writing), the minimum suggested value is 250,000.
WSQUOTA, WSEXTENT	Large enough to avoid excessive page faulting.
FILLM	25 more than the count of database storage areas and snapshot storage areas.

This problem has been corrected in Oracle Rdb7 Release 7.0.6.1. An exceeded quota error during a Detected Asynchronous Prefetch I/O request is handled by simply dismissing the request and not attempting to clean up the buffer pool. The result is that the asynchronous prefetch request is ignored (because the process does not have the quotas required to initiate the I/O), but the desired page will ultimately be read from disk synchronously.

2.1.3 Bugcheck at LCKCCH$LOCK_RET_NOT_OK During Hash Index Creation

Bug 1430433

When the logical name RDMS$CREATE_LAREA_NOLOGGING is defined to "1", certain cases of failure to create a hashed index due to incorrect reserving of storage areas may cause a CREATE INDEX statement to cause a bugcheck dump:

```
$ DEFINE RDMS$CREATE_LAREA_NOLOGGING "1"
$ SQL$
SQL> attach data file 'DUA0:[DB]DB';
SQL> declare transaction read write reserving T1 for exclusive write;
SQL>
SQL> create unique index T1I on T1 (C1, C2, C3) type is HASHED store in A1;
%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>
SQL> create index T2I on T2 (C1) type is HASHED store in A2;
%RDMS-I-BUGCHKDMP, generating bugcheck dump file DUA0:[DB]RDSBUGCHK.DMP;
```

The bugcheck dump contains a call stack similar to the following:

```
$ SEARCH RDSBUGCHK.DMP "EXCEPTION","SAVED PC","-F-","-E-"
***** Exception at 00924930 : LCKCCH$LOCK_RET_NOT_OK + 000000F0
%RDMS-F-NOT_LARDY, area for 54:2:0 not in proper ready mode
Saved PC = 00894358 : DIOFETCH$FETCH_ONE_LINE + 00000208
Saved PC = 00895088 : DIO$FETCH_DBKEY + 000002D8
Saved PC = 00952DC0 : PSI$MODIFY_PCL_ALL + 00000090
Saved PC = 0033E2B4 : SOR$$GET_KEY_PREFIX + 00000A84
Saved PC = 0033E898 : DIODROPDROPMIXEDAREA + 00000308
Saved PC = 008B3330 : DIOLAREA$ERASE_MIXED_LAREA + 00000210
Saved PC = 008B0ED0 : DIOLAREA$DELETE_LAREA + 00000180
Saved PC = 009AB35C : DIOUN$CRLA + 0000057C
Saved PC = 009A9D60 : DIO$UN_DO + 000001C0
Saved PC = 0097661C : RUJUTL$ROLLBACK_LOOP + 000004CC
Saved PC = 00973700 : RUJ$VERB_ROLLBACK + 00000080
Saved PC = 008D84F0 : KOD$VERB_FAILURE + 000000D0
Saved PC = 006C7688 : RDMS$$TOP_DSDI_CLEANUP + 000000A8
Saved PC = 006C70E4 : RDMS$$TOP_DSDI_HNDLR + 00000174
Saved PC = 808A3D94 : symbol not found
Saved PC = 958746BC : symbol not found
***** Exception at 006A83C0 : RDMS$$KOD_INT_READY + 000001D0
%RDMS-F-NOT_LARDY, area for 54:2:0 not in proper ready mode
Saved PC = 006F7748 : RDMS$$EXE_OPEN + 00000E28
Saved PC = 006F6AA0 : RDMS$$EXE_OPEN + 00000180
Saved PC = 006F69D0 : RDMS$$EXE_OPEN + 000000B0
Saved PC = 006F82B4 : RDMS$$EXE_OPEN + 00001994
Saved PC = 0466EAB4 : symbol not found
Saved PC = 0082C688 : RDMS$$INT_START_REQUEST + 00000098
Saved PC = 00470818 : RDMS$$COUNT_RELATION + 000001E8
Saved PC = 004D56A4 : RDMS$$CREATE_INDEX_INFO + 00003F74
Saved PC = 004596B8 : RDMS$$RELEASE_DDL_VM_HNDLR + 00001058
Saved PC = 003C25A4 : BLI$CALLG + 000000BC
Saved PC = 008D55F0 : KOD$SETSTK_AND_CONTINUE + 00000184
```

This problem was caused by the transaction rollback "UNDO" code not correctly accessing a logical area for system records. Unfortunately, the database recovery (DBR) process would fail in the same fashion. In most cases, the process that originally failed would be terminated when the DBR process did not complete correctly.

This problem has been corrected in Oracle Rdb7 Release 7.0.6.1. The logical area for the system records is now readied in the correct mode prior to UNDO processing. The correct error, UNRES_REL, is now returned:

```
SQL> declare transaction read write reserving T1 for exclusive write;
SQL>
SQL> create unique index T1I on T1 (C1, C2, C3) type is HASHED store in A1;
%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>
SQL> create index T2I on T2 (C1) type is HASHED store in A2;
%RDB-E-NO_META_UPDATE, metadata update failed
-RDB-E-UNRES_REL, relation T2 in specified request is not a relation reserved
  in specified transaction
SQL>
```

2.1.4 Attempts to Truncate Snapshot Files Online Hang

Bug 563410

Attempts to truncate snapshot files while there were transactions active in the database would hang until all database transactions ended and those processes did not attempt to start another transaction. The following sequence of events demonstrates the problem. This example uses three different database sessions:

1. Session 1: Start a read only transaction

```
SQL> attach 'file mf_personnel';
SQL> set transaction read only reserving jobs for shared read;
SQL> select * from jobs;
```

2. Session 2: Start another read only transaction

```
SQL> attach 'file mf_personnel';
SQL> set transaction read only reserving jobs for shared read;
SQL> select * from jobs;
```

3. Session 3: Attempt to truncate a snapshot file

```
SQL> alter database file mf_personnel alter storage area jobs
cont> snapshot allocation is 1 pages;
```

4. Session 1: Start another read only transaction

```
SQL> commit;
SQL> set transaction read only reserving jobs for shared read;
SQL> select * from jobs;
```

5. Session 2: Start another read only transaction

```
SQL> commit;
SQL> set transaction read only reserving jobs for shared read;
SQL> select * from jobs;
    .
    .
    .
```

Until both of the transactions committed and refrained from starting another transaction, the truncating process would hang.

This problem has been corrected in Oracle Rdb7 Release 7.0.6.1. Processes will now allow the truncating process to proceed before starting another transaction.

2.1.5 Excessive SPAM Page Locks, I/O and Stalls With Fast Incremental Backup

Bug 754173

When the "fast incremental backup" feature is enabled, processes updating database pages may excessively fetch and lock SPAM pages. At high update rates, this SPAM page lock contention can impact application performance.

A possible workaround to these excessive SPAM fetches is to disable the "fast incremental backup" feature. Use the SQL statement "ALTER DATABASE FILENAME ... NO INCREMENTAL BACKUP SCAN OPTIMIZATION" to disable the "fast incremental backup" feature.

This problem has been reduced in Oracle Rdb7 Release 7.0.6.1. SPAM page fetches for "fast incremental backup" updates have been eliminated in many cases. A bit map of those SPAM pages that have already been evaluated or updated is maintained on each node so that individual processes should not have to check a SPAM page once it has been evaluated on the system.

2.1.6 Date Function Causes RDML/PASCAL Compilation Problems

Bug 908356

A problem in the way date functions were parsed caused errors during RDML compilation.

The following code fragment provides an example of this RDML compilation error.

```
{== bug.rpa ============================}
PROGRAM Personnel (input, output);
DATABASE PERSONNEL = FILENAME 'mf_personnel';
TYPE
    dummy_rec = PACKED RECORD
                    salary_start  : rdml$cddadt_type;
                END;
    dummy_ptr = [unsafe] ^dummy_rec;
VAR
    dum_ptr : dummy_ptr;

FUNCTION dummydate (indate : rdml$cddadt_type) : rdml$cddadt_type;
BEGIN
   dummydate := indate ;
END;

BEGIN
    NEW (dum_ptr);
    READY personnel;
    WITH dum_ptr^ DO
    BEGIN
    FOR sal IN SALARY_HISTORY
        sal.SALARY_START        := dummydate(salary_start);
    END_FOR;
    END;
    COMMIT;
END.

$ rdml /pas bug.rpa
%RDML-E-DMLSYNTAX, Syntax error: found 'SALARY_START' when expecting
'Field name or DB_KEY'
%RDML-I-ATLINE, at line 22 in the file BUG.RPA;
%RDML-I-NODMLOUTPUT, No output file generated due to errors
%RDML-I-SUMMARY, Completed with 1 Errors, 0 warnings, and
                1 informational message
```

A possible workaround for this problem is to change the local variable name so that it is not identical to any column name in the referenced table(s).

Bug 1477955

Another problem in how date functions were parsed by RDML caused errors when optional parameters were omitted from a routine call reference. Although the RDML compilation suceeds, incorrect PASCAL code is generated.

The following code fragment provides an example of this PASCAL compilation error.

```
{== bug.rpa ============================}

PROGRAM Personnel (input, output);
DATABASE PERSONNEL = FILENAME 'MF_PERSONNEL';
VAR
    date1  : rdml$cddadt_type;

FUNCTION dummydate (indate : [TRUNCATE] rdml$cddadt_type) :
rdml$cddadt_type;
BEGIN
   dummydate := indate ;
END;

BEGIN
    READY personnel;
    STORE S IN Salary_history USING
     S.SALARY_START := dummydate;
    END_STORE;

    COMMIT;
END.

$rdml/pascal bug.rpa
$ pascal/nowarn/lis bug

 := dummydate::RDML$CDDADT_TYPE
.............^
%PASCAL-E-IVFUNCALL, Invalid use of function call
-PASCAL-I-NOTBECAST,    - may not be type cast
at line number 204 in file BUG.PAS;1
%PASCAL-E-ENDDIAGS, PASCAL completed with 2 diagnostic
```

There is no workaround for this problem.

These problems have been corrected in Oracle Rdb7 Release 7.0.6.1.

2.1.7 RDBPRE Results in MAXARGEXC Warning from Alpha MACRO Compiler

Bug 1111805

In prior releases of Oracle Rdb7, the RDBPRE pre-processor may generate intermediate code which causes the Alpha MACRO compiler to generate a warning during compilation. For example,

```
Rdb$0013AC6C7569E2919F53FE145::
^
%AMAC-W-MAXARGEXC, MAX_ARGS exceeded in routine
RDB$0013AC6C7569E2919F53FE145, using 3 at line number 167 in file
DISK:[TEST.RDB70]REPRO.MAR;1
```

This warning is generated when an INVOKE statement uses a runtime database name as shown in this example

```
PROGRAM TEST_APPL
    option type = EXPLICIT
    declare string empid, mf_personnel
    empid = "00165"

&Rdb& invoke database DB = filename "mf_personnel"
&Rdb&   runtime filename mf_personnel

&Rdb& declare_stream TEST_STREAM
&Rdb&   using st1 in db.employees
&Rdb&   with st1.employee_id = empid

&Rdb& start_stream TEST_STREAM

&Rdb& fetch TEST_STREAM

&Rdb& end_stream TEST_STREAM
```

```
END PROGRAM
```

These warnings are not serious and the application will continue to work. Oracle recommends defining the following symbol for MACRO whenever RDBPRE is used to compile sources.

```
$ MACRO == "MACRO/WARN=(NOINFO,NOWARN)
```

This problem has been corrected in Oracle Rdb7 Release 7.0.6.1. The RDBPRE pre-processor now correctly counts optional arguments to generated routines that perform the default ready and start transaction actions.

2.1.8 Error Writing File SORTWORK.TMP, Normal Successful Completion

Bug 1495500

Occasionally, a query would fail with the following errors:

```
%RDB-F-SYS_REQUEST, error from system services request
-COSI-F-FILWRITEERR, error writing file <dev-dir>SORTWORKn.TMP;
-SYSTEM-S-NORMAL, normal successful completion
```

While it was obvious that a sort was failing, the reason for the failure was not obvious. The problem was that an IO completion code was being incorrectly tested. The test has been corrected.

This problem has been corrected in Oracle Rdb7 Release 7.0.6.1.

2.1.9 Extraneous Logical Area Created by DROP STORAGE MAP

Bug 703265

In prior releases of Oracle Rdb, the DROP STORAGE MAP statement may create an extra logical area when a simple storage map is deleted. A simple storage map is one that does not specify any STORE clause, i.e. usually only COMPRESSION settings.

This problem occurs because the DROP STORAGE MAP statement assumes that all logical areas for the table have been deleted and adds one to ensure that the table is valid. In general, this problem is harmless but it does consume logical area resources which can only be recovered by rebuilding the database (such as using SQL EXPORT and IMPORT). No data is lost by this command because the DROP STORAGE MAP statement will fail if rows exist in the table.

This problem has been corrected in Oracle Rdb7 Release 7.0.6.1.

2.1.10 Cannot Disable SAME BACKUP FILENAME Clause

Bug 1240826

In prior versions of Oracle Rdb, attempts to remove the setting of SAME BACKUP FILENAME AS JOURNAL clause using NO BACKUP FILENAME were not successful. This clause was only affecting the setting of the related BACKUP FILENAME clause.

This problem has been corrected in Oracle Rdb7 Release 7.0.6.1. The NO BACKUP FILENAME clause can now be used as part of the ALTER DATABASE ... JOURNAL IS ENABLED statement, ALTER JOURNAL clause, or ADD JOURNAL clause to disable the prior (or default) setting of the SAME BACKUP FILENAME attribute.

2.1.11 Query Having OR Compound Predicates With Subquery Returns Wrong Results

Bug 1527102

The following query contains the OR of three predicates: one of which is based on the results of a subquery; one of which is a filter predicate of the form column = literal; and one of which is a constant of the form literal = literal. The query should return 1 row.

```
set flags 'strategy,detail';
select t1.hmcnr from t1 t1
    where t1.ean='5410103914978' and
    (t1.shop_class = (select sho.shop_class from r_shop sho
                      where sho.shop='460')
     or t1.shop_class='A'
     or 'XXX'='460');
Tables:
  0 = t1
  1 = R_SHOP
Cross block of 2 entries
  Cross block entry 1
    Aggregate: (VIA)
    Conjunct: 1.SHOP = '460'
    Conjunct: 'XXX' = '460'
    Get     Retrieval sequentially of relation 1:R_SHOP
  Cross block entry 2
    Conjunct: (0.ean = '5410103914978') AND ((0.shop_class = {subselect}) OR
              (0.shop_class = 'A') OR ('XXX' = '460'))
    Get     Retrieval sequentially of relation 0:t1
HMCNR
 45281
 45134
2 rows selected
```

The key parts of this query which contributed to the situation leading to the error are these:

1. A filter predicate is ANDed to an OR compound predicate
2. The OR compound predicate contains a subquery predicate, a couple of filter predicates and a constant predicate

As a workaround, the query works if the constant predicate is removed.

This problem has been corrected in Oracle Rdb7 Release 7.0.6.1.

2.1.12 Query Using OR/AND Predicates With EXISTS Clause Returns Wrong Results

Bug 1569972

The following query using AND/OR predicates with an EXISTS clause should return 1 row:

```
set flags 'strategy,detail';
```

```
select t1.c1 from t1 t1, t2 t2 where
((t2.c4 = 1 and
  t2.c5 = 5 and
  not exists (select * from t2 t2a                                  <----
              where t2a.c4 = 4 and t2a.c5  = 5)) or                 <----
 (t2.c4  = 4 and t2.c5 = 5))                                        <----
 and t1.c1 = t2.c6                                                  <----
;
Tables:
  0 = T1
  1 = T2
  2 = T2
Cross block of 3 entries
  Cross block entry 1
    Conjunct: {subselect} = 0
      Aggregate-F1: (COUNT-ANY)     Index only retrieval of relation 2:T2
      Index name  T2_H [2:2]
        Key: (2.C4 = 4) AND (2.C5 = 5)
  Cross block entry 2
    Conjunct: (1.C4 = 1) OR (1.C4 = 4)
    Conjunct: 1.C5 = 5
    Conjunct: {subselect} = 0
    Get     Retrieval by index of relation 1:T2
      Index name  T2_H [(2:2)2] Bool
        Key: ((1.C4 = 1) AND (1.C5 = 5)) OR ((1.C4 = 4) AND (1.C5 = 5))
        Bool: 1.C5 = 5
  Cross block entry 3
    Index only retrieval of relation 0:T1
      Index name  T1_H [1:1]
        Key: 0.C1 = 1.C6
0 rows selected
```

The key parts of this query which contributed to the situation leading to the error are these:

1. OR parent predicate with AND predicates on each branch
2. One of the OR branches also includes a subquery, such as NOT EXISTS
3. A second AND predicate is appended after the OR parent predicate

As a workaround, the problem can be corrected if you move the second AND predicate to the front of the OR parent predicate, as follows:

```
set flags 'strategy,detail';

select t1.c1 from t1 t1, t2 t2 where
t1.c1 = t2.c6 and                                                   <----
((t2.c4 = 1 and
  t2.c5 = 5 and
  not exists (select * from t2 t2a                                  <----
              where t2a.c4 = 4 and t2a.c5  = 5)) or                 <----
 (t2.c4  = 4 and t2.c5 = 5))                                        <----
;
```

This problem has been corrected in Oracle Rdb7 Release 7.0.6.1.

2.1.13 Query Using German Collating Sequence Returns Wrong Results

Bug 1530947

The following query, in a database where the German Collating Sequence is used by default, returns wrong results (should return some rows):

```
SELECT  p.datum, p.produkt, p.abtlg, p.stelle
 FROM v_team_datum p,
      produkte g
 where
       p.abtlg=g.abtlg  ;
Conjunct
Match
  Outer loop
    Sort    Conjunct        Aggregate       Sort    Conjunct
    Leaf#01 BgrOnly PROD_DATEN Card=24063
      BgrNdx1 IDX_PROD_DATEN_SORT [1:1] Fan=8
  Inner loop      (zig-zag)
    Conjunct        Get     Retrieval by index of relation PRODUKTE
      Index name  IDX_PRODUKTE_SORT [0:0]
0 rows selected
```

The key parts of this query which contributed to the situation leading to the error are these:

1. The query is a simple join between a view and one table, with the join predicate of CHAR data type
2. The optimizer uses a match strategy to join them, where a comparison of the join keys requires the process of encoding the CHAR data type into the German collating sequence

As a workaround, the query works if a view with the same attributes as the table is used instead of the table itself, as in the following example:

```
SELECT  p.datum, p.produkt, p.abtlg, p.stelle
 FROM v_team_datum p,
      view_produkte g
 where
       p.abtlg=g.abtlg  ;
Cross block of 2 entries
  Cross block entry 1
    Conjunct        Aggregate       Sort    Conjunct
    Leaf#01 BgrOnly PROD_DATEN Card=24063
      BgrNdx1 IDX_PROD_DATEN_SORT [1:1] Fan=8
  Cross block entry 2
    Leaf#02 FFirst PRODUKTE Card=25
      BgrNdx1 IDX_PRODUKTE_SORT [3:3] Fan=6
```

The query works because the optimizer applies a cross strategy instead of a match strategy.

This problem has been corrected in Oracle Rdb7 Release 7.0.6.1.

2.1.14 Left Outer Join Query Returns Wrong Results When ON Clause Evaluates to False

Bug 1581632

The following left outer join query returns wrong results when the join conditions in the ON clause evaluate to false for all rows:

```
set flags 'strategy,detail';
select tt.employee_id, tt.last_name, jh.job_code
from
    (select e.employee_id, e.last_name
     from degrees d, employees e where
        e.employee_id = '00354'
        and d.employee_id = e.employee_id) as tt
    left outer join
    job_history jh
      on tt.last_name = '?' and                          <----
         jh.job_code = tt.employee_id;                   <----
Tables:
  0 = DEGREES
  1 = EMPLOYEES
  2 = JOB_HISTORY
Cross block of 2 entries        (Left Outer Join)
  Cross block entry 1
    Conjunct: "tt.last_name" = '?'
    Merge of 1 entries
      Merge block entry 1
      Cross block of 2 entries
        Cross block entry 1
          Get     Retrieval by index of relation 1:EMPLOYEES
            Index name  EMPLOYEES_HASH [1:1]       Direct lookup
              Key: 1.EMPLOYEE_ID = '00354'
        Cross block entry 2
          Index only retrieval of relation 0:DEGREES
            Index name  DEG_EMP_ID [1:1]
              Key: 0.EMPLOYEE_ID = 1.EMPLOYEE_ID
  Cross block entry 2
    Conjunct: ("tt.last_name" = '?') AND
              (2.JOB_CODE = tt.employee_id)
    Get     Retrieval by index of relation 2:JOB_HISTORY
      Index name  JH_EMPLOYEE_ID [0:0]
0 rows selected
```

The key parts of this query which contributed to the situation leading to the error are these:

1. Left outer join query on a subquery and job_history of mf_personnel database
2. ON clause containing two or more predicates, and the ON clause evaluates to false for all rows, for example:

   ```
   "last_name" = '?' and jh.job_code = tt.employee_id
   ```

There is no known workaround for this problem.

This problem has been corrected in Oracle Rdb7 Release 7.0.6.1.

2.1.15 Query With Two IN Clauses on Two Subqueries Returns Wrong Results

Bug 1585429

The following query with two IN clauses on two subqueries with different match keys, returns a count of 0 when it should return a non-0 count:

```
SELECT count(*) FROM  t1
WHERE
  subclass_id IN (SELECT  DISTINCT subclass_id
                       FROM  t2
                       WHERE  class_id = 'CAJ_C01#')
  AND
  recipe_id   IN (SELECT  recipe_id
                       FROM  t3
                       WHERE  eqp_id = 'CAR-02C'
                      )
;
Aggregate       Conjunct
Match
  Outer loop
    Conjunct
    Match
      Outer loop
        Get     Retrieval by index of relation t1
          Index name  t1_ndx [0:0]
      Inner loop      (zig-zag)
        Aggregate-F1    Conjunct
        Index only retrieval of relation t2
          Index name  t2_ndx [0:0]
  Inner loop      (zig-zag)
    Aggregate-F1    Conjunct        Get
    Retrieval by index of relation t3
      Index name  t3_ndx [1:1]

         0
1 row selected
```

The key parts of this query which contributed to the situation leading to the error are these:

1. Two different IN clauses on two subqueries, with different match keys
2. The query applies a match strategy where the outer leg uses the match key (subclass_id) of another match stream that is different from the other key (recipe_id) of the inner leg without sorting the results of the outer leg using the match key (subclass_id).

Oracle Rdb7 Release 7.0.5 applies a sort node on the outer leg and thus returns the correct results.

As a workaround, use a query outline to change the strategy to cross from match.

This problem has been corrected in Oracle Rdb7 Release 7.0.6.1.

2.1.16 Query Having Same SUBSTRINGs Within CASE Expression Returns Wrong Results

Bugs 1489972, 1485656, 975091

The following queries, containing the same SUBSTRING expressions within a CASE expression, return wrong results.

The following example shows two simple queries (from Bug 1485656 and Bug 975091) having the same subexpression (SUBSTRING) appearing more than once within the CASE expression. The query in the case of Bug 1489972 is more complicated and thus omitted here. It contains unions of several subselect queries with nested views and SUBSTRING/CASE expressions.

```
! Bug 1485656
!  should return the value 1 for the content of y
! ~Xt: Content of y = 1
!
set FLAGS 'TRACE'
declare :x char(2);
declare :y char(1);
begin
set :x='21';
set :y= case
          when  ((substring(:x from 1 for 1)='1') and
                 (substring(:x from 2 for 1)='1') )
                then '0'
          else
                (substring(:x from 2 for 1))
        end;
trace 'Content of y = ', :y ;
end;
The output is:
~Xt: Content of y =

! Bug 975091
!  should return the value of 295 for the column RESP
!
create table t1 (c1 char(12));
insert into t1 value ( '29500000199');

select substring( c1 from 1 for 3) ress,
       case
          when 'a' = 'c' and (substring(c1 from 1 for 3)) = '295'
            then 'a'
          when 'c' =  'c'
            then (substring(c1 from 1 for 3))
          else  ' '
       end resp
  from t1;
RESS   RESP
295
1 row selected
```

The key parts of these queries which contributed to the situation leading to the errors are these:

1. CASE expression contains several similar expressions
2. The expression in the WHEN clause is shared in the same clause of another WHEN clause (in the case of Bug 975091)
3. The expression in the WHEN clause is shared in another part of the CASE statement, such as an ELSE clause (in the case of Bug 1485656)

In the case of Bug 1485656, a workaround is to use an IF instead of a CASE statement to get the correct results:

```
set FLAGS 'TRACE'
declare :x char(2);
declare :y char(1);
begin
set :x='21';
        if  ((substring(:x from 1 for 1)='1') and
             (substring(:x from 2 for 1)='1') )
            then
                set :y='0';
        else
                set :y=(substring(:x from 2 for 1));
        end if;
trace 'Content of y:',:y;
end;
```

Another workaround is to use temporary variables for the substrings.

In the case of Bug 975091, the workaround is to swap the WHEN clauses, as in the following example:

```
select substring( c1 from 1 for 3) ress,
       case
          when 'c' =  'c'
            then (substring(c1 from 1 for 3))                    ! <= 1st
          when 'a' = 'c' and (substring(c1 from 1 for 3)) = '295' ! <= 2nd
            then 'a'
          else  ' '
       end resp
  from t1;
```

This problem has been corrected in Oracle Rdb7 Release 7.0.6.1.

2.1.17 AIJ File Name Was Not Translated When Defined in SQL

Bug 1573242

If a logical name was specified for an AIJ file name, the logical was not being translated at the time the AIJ file was defined using SQL. Since the logical name was stored in the database as the default AIJ file name instead of its translated value, and the default AIJ file name is used to specify the AIJ file name when the AIJ file is created, whenever the logical name was deassigned or changed unexpected results could occur. For example, an RMU/RESTORE executed after the logical name was deassigned created the AIJ file with the same name as the logical name and placed it in the default directory where the database root file was stored since the AIJ file name logical could no longer be translated to a correct AIJ file specification. Now when the AIJ file name is defined in SQL, if it translates to a logical name, the logical name will be translated and the translated file name will be stored in the database, not the logical name. For example, if "tmp1_aij" is specified and this is a logical, "tmp1_aij" will be translated to it's current translation "tmp1:aij_01.aij" which will be stored in the database as the default AIJ filename.

The following example shows that the logical name TMP1_AIJ and not its translated value TMP1:AIJ_01.AIJ was stored in the database when the SQL ALTER DATABASE command was executed.

```
$define tmp1_aij tmp1:aij_01.aij

SQL> alter  database filename my_db journal is enabled
 add journal aij_01 filename tmp1_aij allocation is 500 blocks;

RMU/DUMP/HEADER my_db
```

```
 FILE DSK01:[TMP1]AIJ_01.AIJ;1  (current aij file name)
     FILE TMP1_AIJ              (default aij file name)
```

As a workaround to this problem, do not use a logical name when defining the AIJ file in SQL.

```
SQL> alter  database filename my_db journal is enabled
 add journal aij_01 filename tmp1:aij_01.aij allocation is 500 blocks;

RMU/DUMP/HEADER my_db

 FILE DSK01:[TMP1]AIJ_01.AIJ;1          (current aij file name)
     FILE TMP1:AIJ_01.AIJ;              (default aij file name)
```

This problem has been corrected in Oracle Rdb7 Release 7.0.6.1.

2.1.18 Erroneous RDMS-F-ALSACTIVE Errors

Bug 1613851

The error RDMS-F-ALSACTIVE, "Database replication is active" may be incorrectly returned when attaching to, or performing valid read only transactions on, the Standby Database in a Hot Standby environment. This problem was introduced in Oracle Rdb7 Release 7.0.6.

This problem has been corrected in Oracle Rdb7 Release 7.0.6.1.

2.1.19 Aggregate Query With Nested MIN Function Returns Wrong Results

Bug 1408892

The following query should return the value of ADMN for min(d1.department_ code):

```
create index dept_managerid_code_ndx on departments
    (manager_id,department_code);

select min(d1.department_code),
    min((select min (d2.department_code)
        from departments d2
        where d1.manager_id = d2.manager_id AND
              d2.budget_actual > 0))
    from departments d1;
                  NULL          NULL
1 row selected
```

The key parts of this query which contributed to the situation leading to the error are these:

1. The subselect query has "where" predicates which cause the query to return 0 rows, e.g. "d2.budget_actual > 0"
2. The subselect query contains an aggregate function, e.g. MIN
3. The subselect query is wrapped inside another aggregate function, e.g. MIN

As a workaround to this problem, the query works if the MIN function is removed from the column 'd2.department_code' in the inner subselect, as seen in the following example.

```
select min(d1.department_code),
    min((select d2.department_code
        from departments d2
        where d1.manager_id = d2.manager_id AND
              d2.budget_actual > 0))
    from departments d1;
```

This problem has been corrected in Oracle Rdb7 Release 7.0.6.1.

2.2 SQL Errors Fixed

2.2.1 IMPORT of Multi-file Database as Single File Database May Fail

Bug 1365631

It is possible to EXPORT a multi-file database and then use SQL IMPORT to create a single file database from the interchange file (RBR). This is described in the Rdb documentation and requires the IMPORT statement: to DROP all storage areas (including RDB$SYSTEM) and all storage maps; each HASHED index must be dropped or replaced with a SORTED index; and sorted indices must be redefined to no longer use the STORE clause.

However, if the source database for the EXPORT included reserved storage areas then this type of IMPORT would fail with the following message:

```
%SQL-F-ERRCRESCH, Error creating database filename {databasename}
-RDMS-F-MFDBONLY, operation is not allowed on single-file databases
```

Note

Reserved storage areas can be defined explicitly using the RESERVE ... STORAGE AREAS clause of ALTER and CREATE DATABASE, or implicitly when ALTER DATABASE ... DROP STORAGE AREA clause is used.

This problem has been corrected in Oracle Rdb7 Release 7.0.6.1. The IMPORT command now detects that there are no storage areas being created, and hence that the result database will be single file. The RESERVE STORAGE AREA clause is not imported in this case.

2.2.2 Known Problems With EXPORT and IMPORT Fixed

Bug 1532755

In prior versions of Oracle Rdb7, the EXPORT and IMPORT utilities did not correctly handle some Rdb metadata that was stored in LIST OF BYTE VARYING columns.

1. EXPORT did not correctly save the QUERY HEADER definition for domains and columns.

   EXPORT would introduce a CR/LF delimiter between each segment because it incorrectly treated the query header in the same way as a multi-line text string such as a comment or source definition. The use of this delimiter was introduced in Oracle Rdb7 to better handle large definitions and user comments.

   The workaround for this problem is to ALTER DOMAIN and ALTER TABLE ALTER COLUMN to apply a new and corrected QUERY HEADER to the affected domains and columns. Single segment query headers are not affected by this problem.

2. IMPORT did not correctly handle very long access control lists (ACL) for tables, databases, domains, functions or procedures. The ACL was truncated, usually with a corrupt final entry.

The workaround for this problem is to use RMU/EXTRACT /ITEM=PROTECTION prior to the EXPORT and IMPORT so that the ACL can be reapplied to the database. In this case, a long ACL is one with more than 80 entries. In all likelihood, few databases would encounter this problem.

The EXPORT interchange file (RBR) contains the correct information and so can be used successfully with this and later releases.

These problems have been corrected in Oracle Rdb7 Release 7.0.6.1.

2.2.3 Truncated Values Output by TRACE Statement

Bug 1231207

The TRACE statement did not allocate sufficient space to format large scaled numeric values. When TINYINT, SMALLINT, INTEGER or BIGINT values with non-zero scales were displayed, it was possible for the trailing digit to be truncated. If the dialect was set to SQL92 or ORACLE LEVEL1, then a warning was returned for SQLCODE and SQLSTATE that indicated that a string truncation had occurred.

```
SQL> set dialect 'SQL92';
SQL> attach 'file TEST';
SQL> set flags 'trace';
SQL>
SQL> begin
cont> declare :x integer(1) = -123456789.1;
cont> trace :x;
cont> end;
~Xt: -123456789.
%RDB-I-TRUN_RTRV, string truncated during assignment to a variable or parameter
SQL>
```

This problem has been corrected in Oracle Rdb7 Release 7.0.6.1. TRACE now correctly accounts for the decimal separator in scaled fixed point numeric values.

2.2.4 Multiple NOT NULL Constraints Generate WHYTWICE Exception

Bug 1293766

In prior releases of Oracle Rdb, the CREATE TABLE statement would fail if a NOT NULL clause was applied more than once to a column. This error message did not report the name of the column that caused this error.

```
SQL> create table tttt
cont>     (a integer
cont>           not null not deferrable
cont>           not null deferrable
cont>           unique);
%SQL-F-WHYTWICE, Column is specified more than once in the column list of a
constraint.
SQL>
SQL> show table tttt;
No tables found
SQL>
```

In this example, the NOT NULL is redundant and only one is required for the table. However, to better support tools such as RMU Extract, this error has been made a warning and enhanced to display the name of the column. The CREATE TABLE statement now succeeds.

```
SQL> create table tttt
cont>     (a integer
cont>           not null not deferrable
cont>           not null deferrable
cont>           unique);
%SQL-W-WHYTWICE, Column A is specified more than once in the column list of a
constraint.
SQL>
SQL> show table tttt;
Information for table TTTT

Columns for table TTTT:
Column Name                    Data Type        Domain
-----------                    ---------        ------
    A                          INTEGER
Unique constraint TTTT_UNIQUE_A
Not Null constraint TTTT_NOT_NULL1
Not Null constraint TTTT_A_NOT_NULL
   .
   .
   .
```

The database administrator should review CREATE TABLE statements which produce this warning as the redundant constraint definition may add unneeded overhead to query processing.

This problem has been corrected in Oracle Rdb7 Release 7.0.6.1.

2.2.5 DROP FUNCTION or DROP PROCEDURE Leave Dependency Records

In prior releases of Oracle Rdb7, the DROP FUNCTION and DROP PROCEDURE statements applied to stored routines did not correctly erase old dependency records from RDB$INTERRELATIONS. The result was that actions on these dependent objects would continue to fail, even though no relationship remained with the dropped routine.

The workaround to this problem is to use DROP MODULE which correctly erases these old dependency records.

The following example shows the erroneous action of DROP FUNCTION as well as the workaround that successfully uses DROP MODULE.

```
SQL> create table sample1
cont>    (vector              integer(1),
cont>     pcnt                integer,
cont>     pv_ix               integer,
cont>     patt_ix             integer);
SQL>
SQL> create table sample2
cont>    (tchan               smallint,
cont>     chan_num            smallint,
cont>     mode                char(1),
cont>     pv_ix               integer);
SQL>
SQL> create procedure sethexbits
cont>     (inout char(544) by reference,
cont>      in smallint by value);
cont>     external name sethexbits
cont>         language C
cont>         general parameter style;
SQL>
SQL> create module vecsigmod language SQL
cont>   function vecsig(in :pv_ix integer) returns char(544);
cont>     begin
cont>       declare :hexvecstr char(544) = ' ';
cont>       call sethexbits(:hexvecstr, -1);
cont>       for :x
cont>         as each row of
cont>           select tchan from sample2 where pv_ix = :pv_ix
cont>       do
cont>         call sethexbits(:hexvecstr,:x.tchan);
cont>       end for;
cont>       return :hexvecstr;
cont>     end;
cont> end module;
SQL>
SQL> select cast(rdb$object_name as char(10)) as obj,
cont>        cast(rdb$subobject_name as char(10)) as subobj,
cont>        cast(rdb$entity_name1 as char(10)) as name1,
cont>        cast(rdb$entity_name2 as char(10)) as name2
cont> from rdb$interrelations
cont> where rdb$subobject_name = 'VECSIG'
cont>     or rdb$entity_name2 = 'VECSIG';
 OBJ          SUBOBJ       NAME1        NAME2
              SETHEXBITS   VECSIGMOD    VECSIG
 SAMPLE2                   VECSIGMOD    VECSIG
 SAMPLE2      PV_IX        VECSIGMOD    VECSIG
 SAMPLE2      TCHAN        VECSIGMOD    VECSIG
4 rows selected
SQL>
SQL> drop function vecsig;
SQL>
SQL> select cast(rdb$object_name as char(10)) as obj,
cont>        cast(rdb$subobject_name as char(10)) as subobj,
cont>        cast(rdb$entity_name1 as char(10)) as name1,
cont>        cast(rdb$entity_name2 as char(10)) as name2
cont> from rdb$interrelations
cont> where rdb$subobject_name = 'VECSIG'
cont>     or rdb$entity_name2 = 'VECSIG';
 OBJ          SUBOBJ       NAME1        NAME2
              SETHEXBITS   VECSIGMOD    VECSIG
 SAMPLE2                   VECSIGMOD    VECSIG
 SAMPLE2      PV_IX        VECSIGMOD    VECSIG
 SAMPLE2      TCHAN        VECSIGMOD    VECSIG
4 rows selected
SQL>
SQL> show function vecsig;
```

```
 No Functions Found
SQL>
SQL> show module vecsigmod;
 Module name is: VECSIGMOD
 Header:  vecsigmod language SQL
 No description found
 Owner is:
 Module ID is: 39
 No Procedures Found
 No Functions Found

SQL>
SQL> alter table sample2 drop column tchan;
%RDB-E-NO_META_UPDATE, metadata update failed
-RDMS-F-OBJ_INUSE, object "SAMPLE2.TCHAN" is referenced by
VECSIGMOD.VECSIG (usage: Function)
-RDMS-F-RELFLDNOD, field TCHAN has not been deleted from relation SAMPLE2
SQL>
SQL> drop module vecsigmod;
SQL>
SQL> alter table sample2 drop column tchan;
SQL>
```

This problem has been corrected in Oracle Rdb7 Release 7.0.6.1.

2.3 Oracle RMU Errors Fixed

2.3.1 RMU /UNLOAD /AFTER_JOURNAL Sort Performance

The RMU /UNLOAD /AFTER_JOURNAL command sorts all records for each transaction in order to remove duplicate modifications to the same record within a transaction. Previous versions of the RMU /UNLOAD /AFTER_JOURNAL command used the "SORT32" package to perform this sorting. However, when a large number of transactions are extracted (and particularly when the transactions do not modify many records), the overhead of using the SORT32 package can become significant.

In Oracle Rdb7 Release 7.0.6.1, the RMU /UNLOAD /AFTER_JOURNAL command now utilizes an internal memory-only "Quick Sort" algorithm for transactions that have less than 128 records in the after image journal file. This should result in significant performance improvements for certain cases of extracting data from the after image journal file.

2.3.2 RMU /UNLOAD /AFTER_JOURNAL DBKEY and Records in Mixed Format Storage Areas

Previously, the RMU /UNLOAD /AFTER_JOURNAL command was unable to return the logical area number in the database key (DBKEY) when extracting records stored in a mixed format storage area. The DBKEY would contain a zero for the logical area number. Records extracted from tables stored in uniform format storage areas had the correct DBKEY value because the logical area number is stored in the after-image journal file.

This problem has been corrected in Oracle Rdb7 Release 7.0.6.1. The RMU /UNLOAD /AFTER_JOURNAL command now loads the area inventory pages (AIP) and uses them to translate from a physical DBKEY (as stored in the after-image journal) to a logical DBKEY for each record from a table stored in a mixed format storage area.

2.3.3 Confusing Lock Mode Displays Updated

Previously, Oracle Rdb7 would incorrectly display both requested and granted modes for all lock states.

In the following output from the RMU /SHOW LOCKS command, one lock is on the grant queue and the other lock is on the convert queue. The requested mode is displayed for the granted lock. This is incorrect because a granted lock does not have a request mode; only the granted mode is valid.

```
          -Master Node Info -Lock Mode Information- Remote Node Info-
ProcessID Lock ID  SystemID Requested Granted Queue Lock ID  SystemID
2040A8DB  20009773 00010002 PR        PW      GRANT 20009773 00010002
2040FABC  20004367 00010002 PR        NL      CNVRT 20004367 00010002
2040E672  2000A64A 00010002 EX        NL      WAIT  2000A64A 00010002
```

The impact of this situation has been reduced in Oracle Rdb7 Release 7.0.6.1. Various displays (in the RMU SHOW STATISTICS Utility and the RMU SHOW LOCKS Utility) have been corrected to only display the requested mode value for locks that are in the wait or convert queues. The granted mode value is only displayed for those locks that are in the grant or convert queues. See the OpenVMS System Services Reference Manual for more information on lock queues and modes.

The following output from the RMU /SHOW LOCKS command demonstrates that granted locks do not display a value for the requested mode (it is blank). Only locks on the convert queue display both requested and granted information. Locks on the waiting queue display only the requested mode; the granted mode is blank.

```
          -Master Node Info -Lock Mode Information- Remote Node Info-
ProcessID Lock ID  SystemID Requested Granted Queue Lock ID  SystemID
2040A8DB  20009773 00010002           PW      GRANT 20009773 00010002
2040FABC  20004367 00010002 PR        NL      CNVRT 20004367 00010002
2040E672  2000A64A 00010002 EX                WAIT  2000A64A 00010002
```

2.3.4 RMU/VERIFY Reports PGSPAMENT or PGSPMCLST Errors

Bug 1472061

RMU/VERIFY may sometimes report PGSPAMENT or PGSPMCLST errors when verifying storage areas. These errors indicate that the Space Area Management ("SPAM") page fullness threshold for a particular data page does not match the actual space usage on the data page. For a further discussion of SPAM pages consult the Oracle Rdb7 Guide to Database Maintenance.

Three problems have been found in the Oracle Rdb7 product that may introduce these inconsistencies:

1. In the following situation, the DBR would neglect to update the last SPAM page referenced.
   - The FAST COMMIT feature was enabled
   - A process terminated abnormally and a Database Recovery ("DBR") process ran to recover the failed process
   - The DBR had to "redo" changes made by the failed process
2. An error was sometimes made in the SPAM threshold calculations when there were unused line index ("LDX") entries at the end of the LDX on a data page and the total free space on the page was just below a threshold.

3. When using Row Cache and rows in the cache were deleted or their size changed.

These problems have been corrected in Oracle Rdb7 Release 7.0.6.1. Further introduction of these SPAM inconsistencies should be reduced. Note that existing SPAM errors will remain until manually corrected. Also, while the incidence of these errors has been reduced they cannot be totally eliminated. See Section 5.0.1 for more information.

2.3.5 RMU/SHOW STATISTICS RMS-F-DEV Error With /INPUT

When using a RMU SHOW STATISTICS prerecorded binary file on a different system than the one that originally collected the statistics data, it is possible for the RMU SHOW STATISTICS utility to fail with a "RMS-F-DEV" fatal error as in the following example:

```
$ RMU/SHOW STATISTICS/INPUT=X.DAT
%RMS-F-DEV, error in device name or inappropriate device type for operation
%RMU-F-FATALOSI, Fatal error from the Operating System Interface.
%RMU-F-FTL_SHOW, Fatal error for SHOW operation at 10-JAN-2001 02:47:05.42
```

There is no known workaround for this problem.

This problem has been corrected in Oracle Rdb7 Release 7.0.6.1.

3
Enhancements

3.1 Enhancements Provided in Oracle Rdb7 Release 7.0.6.1

3.1.1 RMU /UNLOAD /AFTER_JOURNAL Database Metadata File

Previously, the RMU /UNLOAD /AFTER_JOURNAL command always required the source database when unloading data from after-image journal files. The database is used to read metadata (information about tables and the physical database structure) required in order to reconstruct records.

As of Oracle Rdb7 Release 7.0.6.1, the RMU /UNLOAD /AFTER_JOURNAL command now supports the ability to read the database and then store a static copy of the data required. This stored copy can then be used in place of the database when executing the RMU /UNLOAD /AFTER_JOURNAL command to unload data. In this way, the original database need not be required when reading the after-image journal files; only the saved metadata information is needed.

Two new qualifiers have been added to the RMU /UNLOAD /AFTER_JOURNAL command: "/SAVE_METADATA = filename" and "/RESTORE_METADATA = filename".

- SAVE_METADATA=filename

  This qualifier indicates that the RMU /UNLOAD /AFTER_JOURNAL command is to write metadata information to the specified file. The RESTORE_METADATA, TABLE and OPTIONS=FILE qualifiers and the AIJ_NAME parameter are not allowed when the SAVE_METADATA qualifier is present. The default file type is ".METADATA".

- RESTORE_METADATA=filename

  This qualifier indicates that the RMU /UNLOAD /AFTER_JOURNAL command is to read database metadata information from the specified file. The DATABASE parameter is required but the database itself is not accessed when the RESTORE_METADATA qualifier is specified. The default file type is ".METADATA".

Because the database is not available when the RESTORE_METADATA qualifier is specified, certain database-specific actions can not be taken. For example, checks for after-image journaling are disabled. And because the static copy of the metadata information is not updated as database structure and table changes are made, it is important to make sure that the metadata file is saved after database DML operations.

When the RESTORE_METADATA=filename qualifier is specified, additional checks are made to ensure that the after-image journal files were created using the same database that was used to create the metadata file. This provides additional security and helps prevent accidental mismatching of files.

The metadata information file used by the RMU /UNLOAD /AFTER_JOURNAL command is in an internal binary format. The contents and format are not documented and are not directly accessible by other utilities. Further, the content and format of the metadata information file is specific to a version of the RMU /UNLOAD /AFTER_JOURNAL utility. As new versions and updates of Oracle Rdb are released, the metadata information file will likely need to re-created. The same version of Rdb must be used to both write and read a metadata information file. The RMU /UNLOAD /AFTER_JOURNAL verifies the format and version of the metadata information file and issues an error message in the case of a version mismatch.

The following example creates a metadata file for the database MFP. This metadata file can be used as input to a later RMU /UNLOAD /AFTER_JOURNAL command.

```
$ RMU /UNLOAD /AFTER_JOURNAL MFP /SAVE_METADATA=MF_MFP.METADATA /LOG
%RMU-I-LMMFWRTCNT, Wrote 107 objects to metadata file
 "DUA0:[DB]MFMFP.METADATA;1"
```

This example uses a previously created metadata information file for the database MFP. The database is not accessed during the unload operation; the database metadata information is read from the file. As the extract operation no longer directly relies on the source database, the AIJ and METADATA files can be moved to another system and extracted there.

```
$ RMU /UNLOAD /AFTER_JOURNAL /RESTORE_METADATA=MF_MFP.METADATA -
    MFP AIJ_BACKUP1 /TABLE=(NAME=TAB1, OUTPUT=TAB1) /LOG
%RMU-I-LMMFRDCNT, Read 107 objects from metadata file
 "DUA0:[DB]MF_MFP.METADATA;1"
%RMU-I-UNLAIJFL, Unloading table TAB1 to DUA0:[DB]TAB1.DAT;1
%RMU-I-LOGOPNAIJ, opened journal file DUA0:[DB]AIJ_BACKUP1.AIJ;1
%RMU-I-AIJRSTSEQ, journal sequence number is "7216321"
%RMU-I-AIJMODSEQ, next AIJ file sequence number will be 7216322
%RMU-I-LOGSUMMARY, total 2 transactions committed
%RMU-I-LOGSUMMARY, total 0 transactions rolled back
------------------------------------------------------------------------
 ELAPSED:  0 00:00:00.15 CPU: 0:00:00.01 BUFIO: 11 DIRIO: 5 FAULTS: 28
Table "TAB1" : 1 record written (1 modify, 0 delete)
Total : 1 record written (1 modify, 0 delete)
```

For debugging purposes, the contents of a metadata information file can be formatted and displayed using the /OPTIONS=DUMP qualifier with the /RESTORE_METADATA qualifier. This dump may be helpful to Oracle Support Engineers during problem analysis. The contents and format of the metadata information file are subject to change.

4
Documentation Corrections

This chapter provides information not currently available in the Oracle Rdb7 documentation set.

4.1 Documentation Corrections

4.1.1 The Halloween Problem

When a cursor is processing rows selected from a table, it is possible that another separate query can interfere with the retrieval of the cursor by modifying the index column's key values used by the cursor.

For instance, if a cursor selects all EMPLOYEES with LAST_NAME >= 'M', it is likely that the query will use the sorted index on LAST_NAME to retrieve the rows for the cursor. If an update occurs during the processing of the cursor which changes the LAST_NAME of an employee from "Mason" to "Rickard", then it is possible that that employee row will be processed twice. First, when it is fetched with name "Mason", and then later when it is accessed by the new name "Rickard".

The Halloween problem is a well known problem in relational databases. Access strategies which optimize the I/O requirements, such as Index Retrieval, can be subject to this problem. Interference from queries by other sessions are avoided by locking and are controlled by the ISOLATION LEVEL options in SQL, or the CONCURRENCY/CONSISTENCY options in RDO/RDML.

Oracle Rdb avoids this problem if it knows that the cursors subject table will be updated. For example, if the SQL syntax UPDATE ... WHERE CURRENT OF is used to perform updates of target rows, or if the RDO/RDML MODIFY statement uses the context variable for the stream. Then the optimizer will choose an alternate access strategy if an update can occur which may cause the Halloween problem. This can be seen in the access strategy in the example below as a "Temporary relation" being created to hold the result of the cursor query.

When you use interactive or dynamic SQL, the UPDATE ... WHERE CURRENT OF or DELETE ... WHERE CURRENT OF statements will not be seen until after the cursor is declared and opened. In these environments, you must use the FOR UPDATE clause to specify that columns selected by the cursor will be updated during cursor processing. This is an indication to the Rdb optimizer so that it protects against the Halloween problem in this case. This is shown in the two examples below Example 4–1 and Example 4–2.

Example 4–1 shows that the EMP_LAST_NAME index is used for retrieval. Any update performed will possibly be subject to the Halloween problem.

Example 4–1 Interactive Cursor with no Halloween Protection

```
SQL> set flags 'strategy';
SQL> declare emp  cursor for
cont> select * from employees where last_name >= 'M'
cont> order by last_name;
SQL> open emp;
Conjunct        Get     Retrieval by index of relation EMPLOYEES
  Index name  EMP_LAST_NAME [1:0]
SQL> close emp;
```

Example 4–2 Interactive Cursor with Halloween Protection

```
SQL> set flags 'strategy';
SQL> declare emp2 cursor for
cont> select * from employees where last_name >= 'M'
cont> order by last_name
cont> for update of last_name;
SQL> open emp2;
Temporary relation      Conjunct        Get
Retrieval by index of relation EMPLOYEES
  Index name  EMP_LAST_NAME [1:0]
SQL> close emp2;
```

Example 4–2 shows that the query specifies that the column LAST_NAME will be updated by some later query. Now the optimizer protects the EMP_LAST_ NAME index used for retrieval by using a "Temporary Relation" to hold the query result set. Any update performed on LAST_NAME will now avoid the Halloween problem.

When you use the SQL precompiler or the SQL module language compiler, it can be determined from usage that the cursor context will possibly be updated during the processing of the cursor because all cursor related statements are present within the module. This is also true for the RDML/RDBPRE precompilers when you use the DECLARE_STREAM and START_STREAM statements and use the same stream context to perform all MODIFY and ERASE statements.

The point to note here is that the protection takes place during the open of the SQL cursor (or RDO stream), not during the subsequent UPDATE or DELETE.

If you execute a separate UPDATE query which modifies rows being fetched from the cursor, then the actual rows fetched will depend upon the access strategy chosen by the Rdb optimizer. As the query is separate from the cursors query (i.e. doesn't reference the cursor context) then the optimizer does not know that the cursor selected rows are potentially updated and so can not perform the normal protection against the Halloween problem.

4.1.2 RDM$BIND_MAX_DBR_COUNT Documentation Clarification

Bug 1495227

The Rdb7 Guide to Database Performance and Tuning Manual, Volume 2, Page A-18, incorrectly describes the use of the RDM$BIND_MAX_DBR_COUNT logical.

Following is an updated description. Note that the difference in actual behavior between what is in the existing documentation and software is that the logical name only controls the number of database recovery processes created at once during "node failure" recovery (that is, after a system or monitor crash or other abnormal shutdown).

When an entire database is abnormally shut down (due, for example, to a system failure), the database will have to be recovered in a "node failure" recovery mode. This recovery will be performed by another monitor in the cluster if the database is opened on another node or will be performed the next time the database is opened.

The RDM$BIND_MAX_DBR_COUNT logical name and the RDB_BIND_MAX_ DBR_COUNT configuration parameter define the maximum number of database recovery (DBR) processes to be simultaneously invoked by the database monitor during a "node failure" recovery.

This logical name and configuration parameter apply only to databases that do not have global buffers enabled. Databases that utilize global buffers have only one recovery process started at a time during a "node failure" recovery.

In a "node failure" recovery situation with the Row Cache feature enabled (regardless of the global buffer state), the database monitor will start a single database recovery (DBR) process to recover the Row Cache Server (RCS) process and all user processes from the oldest active checkpoint in the database.

4.1.3 RMU /UNLOAD /AFTER_JOURNAL NULL Bit Vector Clarification

Each output record from the RMU /UNLOAD /AFTER_JOURNAL command includes a vector (array) of bits. There is one bit for each field in the data record. If a null bit value is 1, the corresponding field is NULL; if a null bit value is 0, the corresponding field is not NULL and contains an actual data value. The contents of a data field that is NULL are not initialized and are not predictable.

The null bit vector begins on a byte boundary. The field RDB$LM_NBV_LEN indicates the number of valid bits (and thus, the number of columns in the table). Any extra bits in the final byte of the vector after the final null bit are unused and the contents are unpredictable.

The following example C program demonstrates one possible way of reading and parsing a binary output file (including the null bit vector) from the RMU /UNLOAD /AFTER_JOURNAL command. This sample program has been tested using Oracle Rdb V7.0.5 and Compaq C V6.2-009 on OpenVMS Alpha V7.2-1. It is meant to be used as a template for writing your own program.

```
/* DATATYPES.C */

#include <stdio.h>
#include <descrip.h>
#include <starlet.h>
#include <string.h>

#pragma member_alignment __save
#pragma nomember_alignment

struct { /* Database key structure */
    unsigned short      lno;    /* line number */
    unsigned int        pno;    /* page number */
    unsigned short      dbid;   /* area number */
    } dbkey;
```

```
typedef struct { /* Null bit vector with one bit for each column */
    unsigned            n_tinyint   :1;
    unsigned            n_smallint  :1;
    unsigned            n_integer   :1;
    unsigned            n_bigint    :1;
    unsigned            n_double    :1;
    unsigned            n_real      :1;
    unsigned            n_fixstr    :1;
    unsigned            n_varstr    :1;
    } nbv_t;

struct { /* LogMiner output record structure for table DATATYPES */
    char                rdb$lm_action;
    char                rdb$lm_relation_name [31];
    int                 rdb$lm_record_type;
    short               rdb$lm_data_len;
    short               rdb$lm_nbv_len;
    __int64             rdb$lm_dbk;
    __int64             rdb$lm_start_tad;
    __int64             rdb$lm_commit_tad;
    __int64             rdb$lm_tsn;
    short               rdb$lm_record_version;
    char                f_tinyint;
    short               f_smallint;
    int                 f_integer;
    __int64             f_bigint;
    double              f_double;
    float               f_real;
    char                f_fixstr[10];
    short               f_varstr_len;   /* length of varchar */
    char                f_varstr[10];   /* data of varchar */
    nbv_t               nbv;
    } lm;

#pragma member_alignment __restore

main ()
{   char timbuf [24];
    struct dsc$descriptor_s dsc = {
        23, DSC$K_DTYPE_T, DSC$K_CLASS_S, timbuf};
    FILE *fp = fopen ("datatypes.dat", "r", "ctx=bin");

    memset (&timbuf, 0, sizeof(timbuf));

    while (fread (&lm, sizeof(lm), 1, fp) != 0)
    {
        printf ("Action     = %c\n",     lm.rdb$lm_action);
        printf ("Table      = %.*s\n",   sizeof(lm.rdb$lm_relation_name),
                                         lm.rdb$lm_relation_name);
        printf ("Type       = %d\n",     lm.rdb$lm_record_type);
        printf ("Data Len   = %d\n",     lm.rdb$lm_data_len);
        printf ("Null Bits  = %d\n",     lm.rdb$lm_nbv_len);

        memcpy (&dbkey, &lm.rdb$lm_dbk, sizeof(lm.rdb$lm_dbk));
        printf ("DBKEY      = %d:%d:%d\n", dbkey.dbid,
                                           dbkey.pno,
                                           dbkey.lno);

        sys$asctim (0, &dsc, &lm.rdb$lm_start_tad, 0);
        printf ("Start TAD  = %s\n", timbuf);

        sys$asctim (0, &dsc, &lm.rdb$lm_commit_tad, 0);
        printf ("Commit TAD = %s\n", timbuf);

        printf ("TSN        = %Ld\n",    lm.rdb$lm_tsn);
        printf ("Version    = %d\n",     lm.rdb$lm_record_version);
```

```
        if (lm.nbv.n_tinyint == 0)
                printf ("f_tinyint  = %d\n", lm.f_tinyint);
        else    printf ("f_tinyint  = NULL\n");

        if (lm.nbv.n_smallint == 0)
                printf ("f_smallint = %d\n", lm.f_smallint);
        else    printf ("f_smallint = NULL\n");

        if (lm.nbv.n_integer == 0)
                printf ("f_integer  = %d\n", lm.f_integer);
        else    printf ("f_integer  = NULL\n");

        if (lm.nbv.n_bigint == 0)
                printf ("f_bigint   = %Ld\n", lm.f_bigint);
        else    printf ("f_bigint   = NULL\n");

        if (lm.nbv.n_double == 0)
                printf ("f_double   = %f\n", lm.f_double);
        else    printf ("f_double   = NULL\n");

        if (lm.nbv.n_real == 0)
                printf ("f_real     = %f\n", lm.f_real);
        else    printf ("f_real     = NULL\n");

        if (lm.nbv.n_fixstr == 0)
                printf ("f_fixstr   = %.*s\n", sizeof (lm.f_fixstr),
                                                lm.f_fixstr);
        else    printf ("f_fixstr   = NULL\n");

        if (lm.nbv.n_varstr == 0)
                printf ("f_varstr   = %.*s\n", lm.f_varstr_len, lm.f_varstr);
        else    printf ("f_varstr   = NULL\n");

        printf ("\n");
    }
}
```

Example sequence of commands to create a table, unload the data and display the contents with this program:

```
SQL> ATTACH 'FILE MF_PERSONNEL';
SQL> CREATE TABLE DATATYPES (
     F_TINYINT TINYINT
    ,F_SMALLINT SMALLINT
    ,F_INTEGER INTEGER
    ,F_BIGINT BIGINT
    ,F_DOUBLE DOUBLE PRECISION
    ,F_REAL REAL
    ,F_FIXSTR CHAR (10)
    ,F_VARSTR VARCHAR (10));
SQL> COMMIT;
SQL> INSERT INTO DATATYPES VALUES (1, NULL, 2, NULL, 3, NULL, 'THIS', NULL);
SQL> INSERT INTO DATATYPES VALUES (NULL, 4, NULL, 5, NULL, 6, NULL, 'THAT');
SQL> COMMIT;
SQL> EXIT;
$ RMU /BACKUP /AFTER_JOURNAL MF_PERSONNEL AIJBCK.AIJ
$ RMU /UNLOAD /AFTER_JOURNAL MF_PERSONNEL AIJBCK.AIJ -
    /TABLE = (NAME=DATATYPES, OUTPUT=DATATYPES.DAT)
$ CC DATATYPES.C
$ LINK DATATYPES.OBJ
$ RUN DATATYPES.EXE
```

4.1.4 Location of Host Source File Generated by the SQL Precompilers

Bug 478898

When the SQL precompiler generates host source files (like .c, .pas, .for) from the precompiler source files, it locates these files based on the /obj qualifier located on the command line given to the SQL precompiler.

The following examples show the location where the host source file is generated. When /obj is not specified on the command line, the object and the host source file take the name of the SQL precompiler source files with the extensions of .obj and .c respectively.

```
LUND> sqlpre/cc scc_try_mli_successful.sc
LUND> dir scc_try_mli_successful.*

Directory MYDISK:[LUND]

SCC_TRY_MLI_SUCCESSFUL.C;1              SCC_TRY_MLI_SUCCESSFUL.OBJ;2
SCC_TRY_MLI_SUCCESSFUL.SC;2

Total of 3 files.
```

When /obj is specified on the command line, the object and the host source take the name given on the qualifier switch. It uses the default of the SQL precompiler source if a filespec is not specified. It uses the defaults of .obj and .c if the extension is not specified. If the host language is other than C, then it uses the appropriate host source extension (like .pas, .for, etc). The files also default to the current directory if a directory spec is not specified.

```
LUND> sqlpre/cc/obj=myobj scc_try_mli_successful.sc
LUND> dir scc_try_mli_successful.*

Directory MYDISK:[LUND]

SCC_TRY_MLI_SUCCESSFUL.SC;2

Total of 1 file.
LUND> dir myobj.*

Directory MYDISK:[LUND]

MYOBJ.C;1           MYOBJ.OBJ;2

Total of 2 files.

LUND> sqlpre/cc/obj=MYDISK:[lund.tmp] scc_try_mli_successful.sc
LUND> dir scc_try_mli_successful.*

Directory MYDISK:[LUND]

SCC_TRY_MLI_SUCCESSFUL.SC;2

Total of 1 file.
LUND> dir MYDISK:[lund.tmp]scc_try_mli_successful.*

Directory MYDISK:[LUND.TMP]

SCC_TRY_MLI_SUCCESSFUL.C;1              SCC_TRY_MLI_SUCCESSFUL.OBJ;2

Total of 2 files.
```

This problem has been corrected in Oracle Rdb7 Release 7.0.6.

4.1.5 Suggestion to Increase GH_RSRVPGCNT Removed

The Oracle Rdb7 for OpenVMS Installation and Configuration Guide contains a section titled "Installing Oracle Rdb Images as Resident on OpenVMS Alpha" that includes information about increasing the value of the OpenVMS system parameter GH_RSRVPGCNT when you modify the RMONSTART.COM or SQL$STARTUP.COM procedures to install Rdb images with the /RESIDENT qualifier.

Note that modifying the parameter GH_RSRVPGCNT is only ever required if the RMONSTART.COM or SQL$STARTUP.COM procedures have been manually modified to install Rdb images with the /RESIDENT qualifier. Furthermore, if the RMONSTART.COM and SQL$STARTUP.COM procedures are executed during the system startup procedure (directly from SYSTARTUP_VMS.COM, for example), then there is no need to modify the GH_RSRVPGCNT parameter.

Oracle and Compaq suggest that you do not modify the value of the GH_ RSRVPGCNT system parameter unless it is absolutely required. Some versions of OpenVMS on some hardware platforms require that GH_RSRVPGCNT be zero in order to ensure the highest level of system performance.

4.1.6 Clarification of the DDLDONOTMIX Error Message

Bug 454080

The ALTER DATABASE statement performs two classes of functions: changing the database root structures in the .RDB file and modifying the system metadata in the RDB$SYSTEM storage area. The first class of changes do not require a transaction to be active. However, the second class requires that a transaction be active. Oracle Rdb does not currently support the mixing of these two classes of ALTER DATABASE clauses.

When you mix clauses that fall into both classes, the error message DDLDONOTMIX "the {SQL-syntax} clause can not be used with some ALTER DATABASE clauses" is displayed, and the ALTER DATABASE statement fails.

```
SQL> alter database filename MF_PERSONNEL
cont> dictionary is not used
cont> add storage area JOB_EXTRA filename JOB_EXTRA;
%RDB-F-BAD_DPB_CONTENT, invalid database parameters in the database parameter
block (DPB)
-RDMS-E-DDLDONOTMIX, the "DICTIONARY IS NOT USED" clause can not be used with
some ALTER DATABASE clauses
```

The following clauses may be mixed with each other but may not appear with other clauses such as ADD STORAGE AREA, or ADD CACHE.

- DICTIONARY IS [NOT] REQUIRED
- DICTIONARY IS NOT USED
- MULTISCHEMA IS { ON | OFF }
- CARDINALITY COLLECTION IS { ENABLED | DISABLED }
- METADATA CHANGES ARE { ENABLED | DISABLED }
- WORKLOAD COLLECTION IS { ENABLED | DISABLED }

If the DDLDONOTMIX error is displayed, then restructure the ALTER DATABASE into two statements, one for each class of actions.

```
SQL> alter database filename MF_PERSONNEL
cont> dictionary is not used;
SQL> alter database filename MF_PERSONNEL
cont> add storage area JOB_EXTRA filename JOB_EXTRA;
```

4.1.7 Compressed Sorted Index Entry Stored in Incorrect Storage Area

This note was originally included in the Oracle Rdb7 Release 7.0.1.3 and 7.0.2 Release Notes. The logical name documented in the note for those releases was documented incorrectly. Below is a corrected note.

In specific cases, in versions V6.1 and V7.0 of Oracle Rdb, when a partitioned, compressed sorted index was created after the data was inserted into the table, b-tree entries may have been inserted into the wrong storage area.

All of the following criteria must be met in order for the possibility of this problem to occur:

- CREATE INDEX is issued after there are records already in the table on which the index is being created
- index must be partitioned over a single column
- index must have compression enabled
- scale factor must be zero on the columns of the index
- no collating sequences specified on the columns of the index
- no descending indexes
- MAPPING VALUES must not be specified

RMU/DUMP/AREA=xx will show that the b-tree entry was not stored in the expected storage area. However, in versions V6.1 and V7.0 of Oracle Rdb, the rows of the table can still be successfully retrieved.

The following example shows the problem:

```
create database
    filename foo
 create storage area Area_1
        filename Area_1
 create storage area Area_2
        filename Area2;

create table T1
        (C1 integer);

! insert data into table prior to index creation
insert into T1 values (0);
commit;
```

```
! create index with COMPRESSION ENABLED
create index Index_1
    on T1 (C1)
    enable compression
    store using (C1)
        in Area_1 with limit of (0)
        otherwise in Area_2;
COMMIT;
!
! Dump out the page for b-tree in AREA_1, there are 0 bytes stored.
! There should be 5 bytes stored for the b-tree entry.
!
RMU/DUMP/AREA=AREA_1
.
.
.                                ....  total B-tree node size: 430
                      0030 2003  0240  line 0 (2:5:0) index: set 48
              002F FFFFFFFF FFFF  0244  owner 47:-1:-1
                           0000  024C  0 bytes of entries <---***** no entry
                           8200  024E  level 1, full suffix
00000000000000000000000000000000  0250  unused '................'
.
.
.
!
! Dump out the page for b-tree in AREA_2, there are 5 bytes stored
!
RMU/DUMP/AREA=AREA_2
.
.
.                                ....  total B-tree node size: 430
                      0031 2003  0240  line 0 (3:5:0) index: set 49
              002F FFFFFFFF FFFF  0244  owner 47:-1:-1
                           000A  024C  10 bytes of entries
                           8200  024E  level 1, full suffix
                          00 05  0250  5 bytes stored, 0 byte prefix <---entry
                     0100008000  0252   key '.....'
                        22B1 10  0257   pointer 47:554:0
.
.
.
```

This problem occurs when index compression is enabled. Therefore, a workaround is to create the index with compression disabled (which is the default). Once this update kit is applied, it is recommended that the index be dropped and recreated with compression enabled to rebuild the b-tree.

Note

In prior versions, the rows were successfully retrieved even though the key values were stored in the wrong storage area. This was due to the range query algorithm skipping empty partitions or scanning extra areas.

However, due to an enhancement in the algorithm for range queries on partitioned SORTED indexes in Oracle Rdb7 Relese 7.0.2, the rows of the table which are stored in the incorrect storage areas may not be retrieved when using the partitioned index.

The optimized algorithm now only scans the relevant index areas (and no longer skips over emtpy areas) resulting in only those rows being returned. Therefore, it is recommended that the index be dropped and re-created. For a short term solution, another alternative is to disable the

new optimization by defining the logical RDMS$INDEX_PART_CHECK to 0.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.3.

4.1.8 Partition Clause is Optional on CREATE STORAGE MAP

Bug 642158

In the *Oracle Rdb7 SQL Reference Manual*, the syntax diagram for the CREATE STORAGE MAP statement incorrectly shows the partition clause as required syntax. The partition clause is not a required clause.

This correction will appear in the next publication of the *Oracle Rdb SQL Reference Manual*.

4.1.9 Oracle Rdb Logical Names

The *Oracle Rdb7 Guide to Database Performance and Tuning* contains a table in Chapter 2 summarizing the Oracle Rdb logical names and configuration parameters. The information in the following table supersedes the entries for the RDM$BIND_RUJ_ALLOC_BLKCNT and RDM$BIND_RUJ_EXTEND_BLKCNT logical names.

Logical Name Configuration Parameter	Function
RDM$BIND_RUJ_ALLOC_BLKCNT	Allows you to override the default value of the .ruj file. The block count value can be defined between 0 and 2 billion with a default of 127.
RDM$BIND_RUJ_EXTEND_BLKCNT	Allows you to pre-extend the .ruj files for each process using a database. The block count value can be defined between 0 and 65535 with a default of 127.

4.1.10 Waiting for Client Lock Message

The *Oracle Rdb7 Guide to Database Performance and Tuning* contains a section in Chapter 3 that describes the Performance Monitor Stall Messages screen. The section contains a list describing the "Waiting for" messages. The description of the "waiting for client lock" message was missing from the list.

A client lock indicates that an Oracle Rdb metadata lock is in use. The term client indicates that Oracle Rdb is a client of the Oracle Rdb locking services. The metadata locks are used to guarantee memory copies of the metadata (table, index, and column definitions) are consistent with the on-disk versions.

The "waiting for client lock" message means the database user is requesting an incompatible locking mode. For example, when trying to delete a table which is in use, the drop operation requests a PROTECTED WRITE lock on the metadata object (such as a table) which is incompatible with the existing PROTECTED READ lock currently used by others of the table.

These metadata locks consist of three longwords. The lock is displayed in text format first, followed by its hexadecimal representation. The text version masks out nonprintable characters with a period (.).

The leftmost value seen in the hexadecimal output contains the ID of the object. The following ID describes the tables, routines, modules and storage map areas.

- For tables and views, the ID represents the unique value found in the RDB$RELATION_ID column of the RDB$RELATIONS system table for the given table.
- For routines, the ID represents the unique value found in the RDB$ROUTINE_ID column of the RDB$ROUTINES system table for the given routine.
- For modules, the ID represents the unique value found in the RDB$MODULE_ID column of the RDB$MODULES system table for the given module.
- For storage map areas, the ID presents the physical area ID. The "waiting for client lock" message on storage map areas is very rare. This may be raised for databases that have been converted from versions prior to Oracle Rdb 5.1.

The next value displayed signifies the object type. The following table describes objects and their hexadecimal type values:

Table 4–1 Object Type Values

Object	Hexadecimal Value
Tables or views	00000004
Routines	00000006
Modules	00000015
Storage map areas	0000000E

The last value in the hexadecimal output represents the lock type. The value 55 indicates this is a client lock.

The following example shows a "waiting for client" lock message from the Stall Messages screen:

```
Process.ID Since......   Stall.reason.............................. Lock.ID.
46001105:2 10:40:46.38 - waiting for client '........' 00000019000000040000005 5
                                                      1        2       3       4
```

The following list describes each part of the client lock:

1 ..... indicates nonprintable characters.

2 00000019 indicates unique identifier hex value 19 (RDB$RELATION_ID = 25).

3 00000004 indicates object type 4 which is a table.

4 00000055 indicates this is a client lock.

To determine the name of the referenced object given the Lock ID the following queries can be used based on the object type:

```
SQL> SELECT RDB$RELATION_NAME FROM RDB$RELATIONS WHERE RDB$RELATION_ID = 25;
SQL> SELECT RDB$MODULE_NAME FROM RDB$MODULES WHERE RDB$MODULE_ID = 12;
SQL> SELECT RDB$ROUTINE_NAME FROM RDB$ROUTINES WHERE RDB$ROUTINE_ID = 7;
```

Note

Because the full client lock output is long, it may require more space than

is allotted for the Stall.reason column and therefore can be overwritten by the Lock.ID. column output.

For more detailed lock information, perform the following steps:

1. Press the L option from the horizontal menu to display a menu of Lock IDs.
2. Select the desired Lock ID.

4.1.11 Documentation Error in *Oracle Rdb7 Guide to Database Performance and Tuning*

The *Oracle Rdb7 Guide to Database Performance and Tuning, Volume 2* contains an error in section C.7, "Displaying Sort Statistics with the R Flag".

When describing the output from this debugging flag, bullet 9 states:

> **Work File Alloc** indicates how many work files were used in the sort operation. A zero (0) value indicates that the sort was accomplished completely in memory.

This is incorrect. This statistic should be described as shown:

> **Work File Alloc** indicates how much space (in blocks) was allocated in the work files for this sort operation. A zero (0) value indicates that the sort was accomplished completely in memory.

This error will be corrected in a future release of *Oracle Rdb Guide to Database Performance and Tuning*.

4.1.12 SET FLAGS Option IGNORE_OUTLINE Not Available

Bug 510968

The *Oracle Rdb7 SQL Reference Manual* described the option IGNORE_ OUTLINE in Table 7-6 of the SET FLAGS section. However, this keyword was not implemented in Oracle Rdb7.

This has been corrected in this release of Oracle Rdb7. This keyword is now recognized by the SET FLAGS statement. As a workaround the logical name RDMS$BIND_OUTLINE_FLAGS "I" can be used to set this attribute.

4.1.13 SET FLAGS Option INTERNALS Not Described

The *Oracle Rdb7 SQL Reference Manual* does not describe the option INTERNALS in Table 7-6 in the SET FLAGS section. This keyword was available in first release of Oracle Rdb7 and is used to enable debug flags output for internal queries such as constraints and triggers. It can be used in conjunction with other options such as STRATEGY, BLR, and EXECUTION. For example, the following flag settings are equivalent to defining the RDMS$DEBUG_FLAGS as ISn and shows the strategy used by the trigge's actions on the AFTER DELETE trigger on the EMPLOYEES table.

```
SQL> SET FLAGS 'STRATEGY, INTERNAL, REQUEST_NAME';
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
```

4.1.14 Documentation for VALIDATE_ROUTINE Keyword for SET FLAGS

The SET FLAGS section of the *Oracle Rdb7 SQL Reference Manual* omitted the description of the VALIDATE_ROUTINE keyword (which can be negated as NOVALIDATE_ROUTINE). This keyword enables the re-validation of an invalidated stored procedure or function. This flag has the same action as the logical RDMS$VALIDATE_ROUTINE described in the *Oracle Rdb7 Guide to Database Performance and Tuning*.

This example shows the re-validation 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 table to be set as valid.

```
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 the SET NOEXECUTE statement in interactive SQL allows the stored routine to be successfully compiled, but it is not executed.

4.1.15 Documentation for Defining the RDBSERVER Logical Name

Bugs 460611 and 563649.

Sections 4.3.7.1 and 4.3.7.2 in the *Oracle Rdb7 for OpenVMS Installation and Configuration Guide* provide the following examples for defining the RDBSERVER logical name:

```
 $ DEFINE RDBSERVER SYS$SYSTEM:RDBSERVER70.EXE
  and
 $ DEFINE RDBSERVER SYS$SYSTEM:RDBSERVER61.EXE
```

These definitions are inconsistent with other command procedures that attempt to reference the RDBSERVERxx.EXE image. The following is one example where the RDBSERVER.COM procedure references SYS$COMMON:<SYSEXE> and SYS$COMMON:[SYSEXE], rather than SYS$SYSTEM:

```
$   if .not. -
       ((f$locate ("SYS$COMMON:<SYSEXE>",rdbserver_image) .ne. log_len) .or. -
        (f$locate ("SYS$COMMON:[SYSEXE]",rdbserver_image) .ne. log_len))
$   then
$       say "''rdbserver_image' is not found in SYS$COMMON:<SYSEXE>"
$       say "RDBSERVER logical is ''rdbserver_image'"
$       exit
$   endif
```

In this case, if the logical name were defined as instructed in the *Oracle Rdb7 for OpenVMS Installation and Configuration Guide*, the image would not be found.

The *Oracle Rdb7 for OpenVMS Installation and Configuration Guide* should define the logical name as follows:

```
DEFINE RDBSERVER SYS$COMMON:<SYSEXE>RDBSERVER70.EXE
 and
DEFINE RDBSERVER SYS$COMMON:<SYSEXE>RDBSERVER61.EXE
```

4.1.16 Undocumented SET Commands and Language Options

The following SET statements were omitted from the Oracle Rdb7 documentation.

4.1.16.1 QUIET COMMIT Option

The SET QUIET COMMIT statement (for interactive and dynamic SQL), the module header option QUIET COMMIT, the /QUIET_COMMIT (and /NOQUIET_COMMIT) qualifier for SQL module language, or the /SQLOPTIONS=QUIET_COMMIT (and NOQUIET_COMMIT) option for the SQL language precompiler allows the programmer to control the behavior of the COMMIT and ROLLBACK statements in cases where there is no active transaction.

By default, if there is no active transaction, SQL will raise an error when COMMIT or ROLLBACK is executed. This default is retained for backward compatibility for applications that may wish to detect the situation. If QUIET COMMIT is set to ON, then a COMMIT or ROLLBACK executes successfully when there is no active transaction.

Note

Within a compound statement, the COMMIT and ROLLBACK statements in this case are ignored.

Examples

In interactive or dynamic SQL, the following SET command can be used to disable or enable error reporting for COMMIT and ROLLBACK when no transaction is active. The parameter to the SET command is a string literal or host variable containing the keyword ON or OFF. The keywords may be in any case (upper, lower, or mixed).

```
SQL> COMMIT;
%SQL-F-NO_TXNOUT, No transaction outstanding
SQL> ROLLBACK;
%SQL-F-NO_TXNOUT, No transaction outstanding
SQL> SET QUIET COMMIT 'on';
SQL> ROLLBACK;
SQL> COMMIT;
SQL> SET QUIET COMMIT 'off';
SQL> COMMIT;
%SQL-F-NO_TXNOUT, No transaction outstanding
```

In the SQL module language or precompiler header, the clause QUIET COMMIT can be used to disable or enable error reporting for COMMIT and ROLLBACK when no transaction is active. The keyword ON or OFF must be used to enable or disable this feature. The following example enables QUIET COMMIT so that no error is reported if a COMMIT is executed when no transaction is active. For example:

```
MODULE TXN_CONTROL
LANGUAGE BASIC
PARAMETER COLONS
QUIET COMMIT ON

PROCEDURE S_TXN (SQLCODE);
SET TRANSACTION READ WRITE;

PROCEDURE C_TXN (SQLCODE);
COMMIT;
```

4.1.16.2 COMPOUND TRANSACTIONS Option

The SET COMPOUND TRANSACTIONS statement (for interactive and dynamic SQL) and the module header option COMPOUND TRANSACTIONS allows the programmer to control the SQL behavior for starting default transactions for compound statements.

By default, if there is no current transaction, SQL will start a transaction before executing a compound statement or stored procedure. However, this may conflict with the actions within the procedure, or may start a transaction for no reason if the procedure body does not perform any database access. This default is retained for backward compatibility for applications that may expect a transaction to be started for the procedure.

If COMPOUND TRANSACTIONS is set to EXTERNAL, then SQL starts a transaction before executing the procedure; otherwise, if it is set to INTERNAL, it allows the procedure to start a transaction as required by the procedure execution.

Examples

In interactive or dynamic SQL, the following SET command can be used to disable or enable transactions started by the SQL interface. The parameter to the SET command is a string literal or host variable containing the keyword INTERNAL or EXTERNAL. The keywords may be in any case (upper, lower, or mixed). For example:

```
SQL> SET COMPOUND TRANSACTIONS 'internal';
SQL> CALL START_TXN_AND_COMMIT ();
SQL> SET COMPOUND TRANSACTIONS 'external';
SQL> CALL UPDATE_EMPLOYEES (...);
```

In the SQL module language or precompiler header, the clause COMPOUND TRANSACTIONS can be used to disable or enable starting a transaction for procedures. The keyword INTERNAL or EXTERNAL must be used to enable or disable this feature.

```
MODULE TXN_CONTROL
LANGUAGE BASIC
PARAMETER COLONS
COMPOUND TRANSACTIONS INTERNAL

PROCEDURE S_TXN (SQLCODE);
BEGIN
SET TRANSACTION READ WRITE;
END;
```

```
PROCEDURE C_TXN (SQLCODE);
BEGIN
COMMIT;
END;
```

4.1.17 Undocumented Size Limit for Indexes with Keys Using Collating Sequences

Bug 586079

When a column is defined with a collating sequence, the index key is specially encoded to incorporate the correct ordering (collating) information. This special encoding takes more space than keys encoded for ASCII (the default when no collating sequence is used). Therefore, the encoded string uses more than the customary one byte per character of space within the index. This is true for all versions of Oracle Rdb that support collating sequences.

For all collating sequences, except Norwegian, the space required is approximately 9 bytes for every 8 characters. So, a CHAR (24) column will require approximately 27 bytes. For Norwegian collating sequences, the space required is approximately 10 bytes for every 8 characters.

The space required for encoding the string must be taken into account when calculating the size of an index key against the limit of 255 bytes. Suppose a column defined with a collating sequence of GERMAN was used in an index. The length of that column is limited to a maximum of 225 characters because the key will be encoded in 254 bytes.

The following example demonstrates how a 233 character column, defined with a German collating sequence and included in an index, exceeds the index size limit of 255 bytes, even though the column is defined as less than 255 characters in length:

```
SQL> CREATE DATABASE
cont>      FILENAME 'TESTDB.RDB'
cont>      COLLATING SEQUENCE GERMAN GERMAN;
SQL> CREATE TABLE EMPLOYEE_INFO (
cont>      EMP_NAME CHAR (233));
SQL> CREATE INDEX EMP_NAME_IDX
cont>      ON EMPLOYEE_INFO (
cont>      EMP_NAME     ASC)
cont>      TYPE IS SORTED;
%RDB-E-NO_META_UPDATE, metadata update failed
-RDMS-F-INDTOOBIG, requested index is too big
```

4.1.18 Changes to RMU/REPLICATE AFTER/BUFFERS Command

The behavior of the RMU/REPLICATE AFTER/BUFFERS command has been changed. The /BUFFERS qualifier may be used with either the CONFIGURE option or the START option.

When using local buffers, the AIJ log roll-forward server (LRS) will use a minimum of 4096 buffers. The value provided to the /BUFFERS qualifier will be accepted, but it will be ignored if it is less than 4096. In addition, further parameters will be checked and the number of buffers may be increased if the resulting calculations are greater than the number of buffers specified by the /BUFFERS qualifier. If the database is configured to use more than 4096 AIJ request blocks (ARBs), then the number of buffers may be increased to the number of ARBs configured for the database. The LRS ensures that there are at least 10 buffers for every possible storage area in the database. Thus, if the total

number of storage areas (both used and reserved) multiplied by 10 results in a greater number of buffers, that number will be used.

When global buffers are used, the number of buffers used by the AIJ log roll-forward server is determined as follows:

- If the /BUFFERS qualifier is omitted and the /ONLINE qualifier is specified, the number of buffers will default to the previously configured value, if any, or 256, whichever is larger.
- If the /BUFFERS qualifier is omitted and the /ONLINE qualifier is not specified or the /NOONLINE is specified, the number of buffers will default to the maximum number of global buffers allowed per user ("USER LIMIT"), or 256, whichever is larger.
- If the /BUFFERS qualifier is specified, that value must be at least 256, and it may not be greater than the maximum number of global buffers allowed per user ("USER LIMIT").

The /BUFFER qualifier now enforces a minimum of 256 buffers for the AIJ log roll-forward server. The maximum number of buffers allowed is still 524288 buffers.

4.1.19 Change in the Way RDMAIJ Server is Set Up in UCX

Starting with Oracle Rdb V7.0.2.1, the RDMAIJ image has become a varianted image. Therefore, the information in section 2.12, "Step 10: Specify the Network Transport Protocol," of the *Oracle Rdb7 and Oracle CODASYL DBMS Guide to Hot Standby Databases* has become outdated in regards to setting up the RDMAIJSERVER object when using UCX as the network transport protocol. The UCX SET SERVICE command should now look similar to the following:

```
$ UCX SET SERVICE RDMAIJ -
      /PORT=<port_number> -
      /USER_NAME=RDMAIJ -
      /PROCESS_NAME=RDMAIJ -
      /FILE=SYS$SYSTEM:RDMAIJSERVER.com -
      /LIMIT=<limit>
```

And for Oracle Rdb multiversion, it should look similar to the following:

```
$ UCX SET SERVICE RDMAIJ70 -
      /PORT=<port_number> -
      /USER_NAME=RDMAIJ70 -
      /PROCESS_NAME=RDMAIJ70 -
      /FILE=SYS$SYSTEM:RDMAIJSERVER70.com -
      /LIMIT=<limit>
```

The installation procedure for Oracle Rdb creates a user named RDMAIJ(nn) and places a file called RDMAIJSERVER(nn).com in SYS$SYSTEM and the RMONSTART(nn).COM command procedure will try to enable a service called RDMAIJ(nn) if UCX is installed and running.

Changing the RDMAIJ server to a multivarianted image does not impact installations using DECNet since the correct DECNet object is created during the Rdb installation.

4.1.20 CREATE INDEX Supported for Hot Standby

On page 1-13 of the *Guide to Hot Standby Databases*, the add new index operation is incorrectly listed as an offline operation not supported by Hot Standby. The CREATE INDEX operation is now fully supported by Hot Standby, as long as the transaction does not span all available AIJ journals, including emergency AIJ journals.

4.1.21 Dynamic OR Optimization Formats

Bug 711643

In Table C-2 on Page C-7 of the *Oracle Rdb7 Guide to Database Performance and Tuning*, the dynamic OR optimization format is incorrectly documented as [l:h...]n. The correct formats for Oracle Rdb Release 7.0 and later are [(l:h)n] and [l:h,l2:h2].

5
Known Problems and Restrictions

This chapter describes problems, restrictions, and workarounds known to exist in Oracle Rdb7 Release 7.0.6.1.

5.0.1 RMU/VERIFY Reports PGSPAMENT or PGSPMCLST Errors

RMU/VERIFY may sometimes report PGSPAMENT or PGSPMCLST errors when verifying storage areas. These errors indicate that the Space Area Management ("SPAM") page fullness threshold for a particular data page does not match the actual space usage on the data page. For a further discussion of SPAM pages, consult the Oracle Rdb7 Guide to Database Maintenance.

In general, these errors will not cause any adverse affect on the operation of the database. There is potential for space on the data page to not be totally utilized, or for a small amount of extra I/O to be expended when searching for space in which to store new rows. But unless there are many of these errors then the impact should be negligible.

It is possible for these inconsistencies to be introduced by errors in the Oracle Rdb7 product. When those cases are discovered, Oracle Rdb7 is corrected to prevent the introduction of the inconsistencies. It is also possible for these errors to be introduced during the normal operation of the product. The following scenario can leave the SPAM pages inconsistent:

1. A process inserts a row on a page, and updates the threshold entry on the corresponding SPAM page to reflect the new space utilization of the data page. The data page and SPAM pages are not flushed to disk.
2. Another process notifies the first process that it would like to access the SPAM page being held by the process. The first process flushes the SPAM page changes to disk and releases the page. Note that it has not flushed the data page.
3. The first process then terminates abnormally (for example, from the DCL STOP/IDENTIFICATION command). Since that process never flushed the data page to disk, it never wrote the changes to the Recovery Unit Journal (RUJ) file. Since there were no changes in the RUJ file for that data page, then the Database Recovery ("DBR") process did not need to rollback any changes to the page. The SPAM page retains the threshold update change made above even though the data page was never flushed to disk.

While it would be possible to create mechanisms to ensure that SPAM pages do not become out of synch with their corresponding data pages, the performance impact would not be trivial. Since these errors are relatively rare and the impact is not significant, the introduction of these errors is considered to be part of the normal operation of the Oracle Rdb7 product. If it can be proven that the errors are not due to the scenario above then Oracle Product Support should be contacted.

PGSPAMENT and PGSPMCLST errors may be corrected by doing any one of the following operations:

- Recreate the database by performing:
  1. SQL EXPORT
  2. SQL DROP DATABASE
  3. SQL IMPORT
- Recreate the database by performing:
  1. RMU/BACKUP
  2. SQL DROP DATABASE
  3. RMU/RESTORE
- Repair the SPAM pages by using the RMU/REPAIR command. Note that the RMU/REPAIR command does not write its changes to an after-image journal (AIJ) file. Therefore, Oracle recommends that a full database backup be performed immediately after using the RMU/REPAIR command.

5.0.2 Behavior Change in 'With System Logical_Name Translation' Clause

The way logical name translation is performed when 'with system logical_name translation' is specified in the 'location' clause of the 'create function' or the 'create routine' statements has changed. This change occured between VAX/VMS V5.5-2 and OpenVMS V7.1.

When 'with system logical_name translation' is specified, any logical name in the location string is expanded using only EXECUTIVE_MODE logical names. In VAX/VMS V5.5-2, the logical names are expanded from the SYSTEM logical name table only. In OpenVMS V7.1, the logical names are expanded from the first definition found when searching the logical name tables in (LNM$FILE_DEV) order.

Thus, if a logical is only defined in the EXECUTIVE_MODE SYSTEM table (and in no other EXECUTIVE_MODE tables), then there will be no apparent change in behavior. However, if a logical name has been defined in the EXECUTIVE_ MODE GROUP table and in the EXECUTIVE_MODE SYSTEM table, then on VAX/VMS V5.5 the SYSTEM table translation will be used and on OpenVMS V7.1 the GROUP table translation will be used.

Oracle believes that this behavioral change is still in keeping with the secure intent of this clause for external routines. An OpenVMS user must have SYSNAM privilege to define an EXEC mode logical in any table. Therefore, it still provides a secure method of locating production sharable images for use by the Rdb server.

A future version of the Oracle Rdb SQL Reference manual will be reworded to remove the reference to the SYSTEM logical name table in the description. The keyword SECURE will be synonymous with SYSTEM in this context.

As an example, if the logical TEST_EXTRTN_1 is defined as:

```
$ show logical/access_mode=executive_mode test_extrtn_1
   "TEST_EXTRTN_1" = "NOSUCHIMG9" (LNM$PROCESS_TABLE)
   "TEST_EXTRTN_1" = "NOSUCHIMGA" (LNM$JOB_9D277AC0)
   "TEST_EXTRTN_1" = "NOSUCHIMGB" (TEST$GROUP_LOGICALS)
   "TEST_EXTRTN_1" = "DISK1:[TEST]EXTRTN.EXE" (LNM$SYSTEM_TABLE)
```

Then under VAX/VMS V5.5-2, TEST_EXTRTN_1 will be translated as "DISK1:[TEST]EXTRTN.EXE" whereas under OpenVMS V7.1 it will be translated as "NOSUCHIMG9".

5.0.3 Carry-Over Locks and NOWAIT Transactions Clarification

In NOWAIT transactions, the BLAST mechanism cannot be used. For the blocking user to receive the BLAST signal, the requesting user must request the locked resource with WAIT (which a NOWAIT transaction does not do). Oracle Rdb defines a resource called NOWAIT, which is used to indicate that a NOWAIT transaction has been started. When a NOWAIT transaction starts, the user requests the NOWAIT resource. All other database users hold a lock on the NOWAIT resource so that when the NOWAIT transaction starts, all other users are notified with a NOWAIT BLAST. The BLAST causes blocking users to release any carry-over locks. There can be a delay before the transactions with carry-over locks detect the presence of the NOWAIT transaction and release their carry-over locks. You can detect this condition by examining the stall messages. If the "Waiting for NOWAIT signal (CW)" stall message appears frequently, then the application is probably experiencing a decrease in performance and you should consider disabling the carry-over lock behavior.

5.0.4 Strict Partitioning May Scan Extra Partitions

When you use a WHERE clause with the less than (<) or greater than (>) operator and a value that is the same as the boundary value of a storage map, Oracle Rdb7 scans extra partitions. A boundary value is a value specified in the WITH LIMIT OF clause. The following example, executed while the logical name RDMS$DEBUG_FLAGS is defined as "S", illustrates the behavior:

```
ATTACH 'FILENAME MF_PERSONNEL';
CREATE TABLE T1 (ID INTEGER, LAST_NAME CHAR(12), FIRST_NAME CHAR(12));
CREATE STORAGE MAP M FOR T1 PARTITIONING NOT UPDATABLE
 STORE USING (ID)
 IN EMPIDS_LOW WITH LIMIT OF (200)
 IN EMPIDS_MID WITH LIMIT OF (400)
 OTHERWISE IN EMPIDS_OVER;
INSERT INTO T1 VALUES (150,'Boney','MaryJean');
INSERT INTO T1 VALUES (350,'Morley','Steven');
INSERT INTO T1 VALUES (300,'Martinez','Nancy');
INSERT INTO T1 VALUES (450,'Gentile','Russ');

SELECT * FROM T1 WHERE ID > 400;
Conjunct        Get     Retrieval sequentially of relation T1
Strict Partitioning: part       2       3
          ID   LAST_NAME      FIRST_NAME
         450   Gentile        Russ
1 row selected
```

In the previous example, partition 2 does not need to be scanned. This does not affect the correctness of the result. Users can avoid the extra scan by using values other than the boundary values.

5.0.5 Exclusive Access Transactions May Deadlock With RCS Process

If a table is frequently accessed by long running transactions that request READ /WRITE access reserving the table for EXCLUSIVE WRITE, and if the table has one or more indexes, you may experience deadlocks between the user process and the Row Cache Server (RCS) process.

There are at least three suggested workarounds to this problem:

1. Reserve the table for SHARED WRITE.

2. Close the database and disable row cache for the duration of the exclusive transaction.
3. Change the checkpoint interval for the RCS process to a time longer than the time required to complete the batch job and then trigger a checkpoint just before the batch job starts. Set the interval back to a smaller interval after the checkpoint completes.

5.0.6 Oracle Rdb and OpenVMS ODS-5 Volumes

The OpenVMS Version 7.2 release introduced Extended File Specifications, which consists of two major components:

- A new, optional, volume structure, ODS-5, which provides support for file names that are longer and have a greater range of legal characters than in previous versions of OpenVMS
- Support for deep directories

ODS-5 was introduced primarily to provide enhanced file sharing capabilities for users of Advanced Server for OpenVMS 7.2 (formerly known as PATHWORKS for OpenVMS), as well as DCOM and JAVA applications.

In some cases, Oracle Rdb performs its own file and directory name parsing and explicitly requires ODS-2 (the traditional OpenVMS volume structure) file and directory name conventions to be followed. Because of this knowledge, Oracle does not support any Oracle Rdb database file components (including root files, storage area files, after image journal files, record cache backing store files, database backup files, after image journal backup files, etc.) that utilize any non-ODS-2 file naming features. For this reason, Oracle recommends that Oracle Rdb database components not be located on ODS-5 volumes.

A future release of Oracle Rdb is expected to relax some of these restrictions and support ODS-5 volumes.

5.0.7 Clarification of the USER Impersonation Provided by the Oracle Rdb Server

Bug 551240

In Oracle Rdb V6.1, a new feature was introduced which allowed a user to attach (or connect) to a database by providing a username (USER keyword) and a password (USING keyword). This functionality allows the Rdb Server to impersonate those users in two environments.

- Remote Database Access. When DECnet is used as the remote transport, the Rdb/Dispatch layer of Oracle Rdb uses the provided username and password, or proxy access to create a remote process which matches the named user. However, in a remote connection over TCP/IP, the RDBSERVER process is always logged into RDB$REMOTE rather than a specified user account. In this case the Rdb Server impersonates the user by using the user's UIC (user identification code) during privilege checking. The UIC is assigned by the OpenVMS AUTHORIZE utility.
- SQL/Services database class services. When SQL/Services (possibly accessed by ODBC) accesses a database, it allows the user to logon to the database and the SQL/Services server then impersonates that user in the database.

When a database has access control established using OpenVMS rights identifiers, then access checking in these two environments does not work as expected. For example, if a user JONES was granted the rights identifier PAYROLL_ACCESS, then you would expect a table in the database with SELECT access granted to PAYROLL_ACCESS to be accessible to JONES. This does not currently work because the Rdb Server does not have the full OpenVMS security profile loaded, just the UIC. So only access granted to JONES is allowed.

This problem results in an error being reported such as the following from ODBC:

```
[Oracle][ODBC][Rdb]%RDB-E-NO_PRIV privileged by database facility (#-1028)
```

This is currently a restriction in this release of Oracle Rdb. In the next major release, support will be provided to inherit the users full security profile into the database.

5.0.8 Index STORE Clause WITH LIMIT OF Not Enforced in Single Partition Map

Bug 413410

An index which has a STORE clause with a single WITH LIMIT OF clause and no OTHERWISE clause doesn't validate the inserted values against the high limit. Normally values beyond the last WITH LIMIT OF clause are rejected during INSERT and UPDATE statements.

Consider this example:

```
create table PTABLE (
    NR
        INTEGER,
    A
        CHAR (2));
create index NR_IDX
    on PTABLE (
    NR)
    type is HASHED
    store using (NR)
        in EMPIDS_LOW
            with limit of (10);
```

When a value is inserted for NR that exceeds the value 10, then an error such as "%RDMS-E-EXCMAPLIMIT, exceeded limit on last partition in storage map for NR_IDX" should be generated. However, this error is only reported if the index has two or more partitions.

A workaround for this problem is to create a CHECK constraint on the column to restrict the upper limit. e.g. CHECK (NR <= 10). This check constraint should be defined as NOT DEFERRABLE and will be solved using an index lookup.

This problem will be corrected in a future version of Oracle Rdb.

5.0.9 Unexpected NO_META_UPDATE Error Generated by DROP MODULE ... CASCADE When Attached by PATHNAME

Bug 755182

The SQL statement DROP MODULE ... CASCADE may sometimes generate an unexpected NO_META_UPDATE error. This occurs when the session attaches to a database by PATHNAME.

```
SQL> drop module m1 cascade;
%RDB-E-NO_META_UPDATE, metadata update failed
-RDMS-F-OBJ_INUSE, object "M1P1" is referenced by M2.M2P1 (usage: Procedure)
-RDMS-E-MODNOTDEL, module "M1" has not been deleted
```

This error occurs because the CASCADE option is ignored because the Oracle CDD/Repository does not support CASCADE. The workaround is to attach by FILENAME and perform the metadata operation.

In a future version of Oracle Rdb, an informational message will be issued describing the downgrade from CASCADE to RESTRICT in such cases.

5.0.10 Unexpected DATEEQLILL Error During IMPORT With CREATE INDEX or CREATE STORAGE MAP

Bug 1094071

When the SQL IMPORT statement includes CREATE STORAGE MAP or CREATE INDEX statements which use TIMESTAMP or DATE ANSI literals in the WITH LIMIT OF clause, it fails with the following error:

```
%SQL-F-UNSDATXPR, Unsupported date expression
-SQL-F-DATEEQLILL, Operands of date/time comparison are incorrect
```

The same CREATE STORAGE MAP or CREATE INDEX statements work correctly when used outside of the IMPORT statement.

This error is generated because the SQL IMPORT statement tries to validate the data type of the column against that of the literal value. However, during this phase of the IMPORT, the table does not yet exist.

A workaround for this problem is to use DATE VMS literals in the WITH LIMIT OF clause and allow the Rdb Server to perform the data type conversion at runtime.

This restriction will be relaxed in a future version of Oracle Rdb.

5.0.11 Application and Oracle Rdb Both Using SYS$HIBER

In application processes that use Oracle Rdb and the $HIBER system service (possibly via RTL routines such as LIB$WAIT), it is important that the application ensures that the event being waited for has actually occurred. Oracle Rdb uses $HIBER/$WAKE sequences for interprocess communications particularly when the ALS (AIJ Log Server) or the Row Cache features are enabled.

Oracle Rdb's use of the $WAKE system service can interfere with other users of $HIBER (such as the routine LIB$WAIT) that do not check for event completion, possibly causing a $HIBER to be unexpectedly resumed without waiting at all.

To avoid these situations, consider altering the application to use a code sequence that avoids continuing without a check for the operation (such as a delay or a timer firing) being complete.

The following pseudo-code shows one example of how a flag can be used to indicate that a timed-wait has completed correctly. The wait does not complete until the timer has actually fired and set TIMER_FLAG to TRUE. This code relies on ASTs being enabled.

```
ROUTINE TIMER_WAIT:
    BEGIN
    ! Clear the timer flag
    TIMER_FLAG = FALSE

    ! Schedule an AST for sometime in the future
    STAT = SYS$SETIMR (TIMADR = DELTATIME, ASTRTN = TIMER_AST)
    IF STAT <> SS$_NORMAL THEN LIB$SIGNAL (STAT)

    ! Hibernate.  When the $HIBER completes, check to make
    ! sure that TIMER_FLAG is set indicating that the wait
    ! has finished.
    WHILE TIMER_FLAG = FALSE
    DO  SYS$HIBER()
    END

ROUTINE TIMER_AST:
    BEGIN
    ! Set the flag indicating that the timer has expired
    TIMER_FLAG = TRUE

    ! Wake the main-line code
    STAT = SYS$WAKE ()
    IF STAT <> SS$_NORMAL THEN LIB$SIGNAL (STAT)
    END
```

Starting with OpenVMS V7.1, the LIB$WAIT routine has been enhanced via the FLAGS argument (with the LIB$K_NOWAKE flag set) to allow an alternate wait scheme (using the $SYNCH system service) that can avoid potential problems with multiple code sequences using the $HIBER system service. See the OpenVMS RTL Library (LIB$) Manual for more information about the LIB$WAIT routine.

5.0.12 IMPORT Unable to Import Some View Definitions

Bug 520651

View definitions that reference SQL functions, that is functions defined by the CREATE MODULE statement, cannot currently be imported by the SQL IMPORT statement. This is because the views are defined before the functions themselves exist.

The following example shows the errors from IMPORT.

```
IMPORTing view TVIEW
%SQL-F-NOVIERES, unable to import view TVIEW
%RDB-E-NO_META_UPDATE, metadata update failed
-RDB-E-OBSOLETE_METADA, request references metadata objects that no
longer exist
-RDMS-E-RTNNEXTS, routine FORMAT_OUT does not exist in this database
%RDB-E-OBSOLETE_METADA, request references metadata objects that no
longer exist
-RDMS-F-TABNOTDEF, relation TVIEW is not defined in database
```

The following script can be used to demonstrate the problem.

```
create database filename badimp;
create table t (sex char);

create module TFORMAT
    language SQL
```

```
    function FORMAT_OUT (:s char)
    returns char(4);
    return (case :s
            when 'F' then 'Female'
            when 'M' then 'Male'
            else NULL
            end);
end module;

create view TVIEW (m_f) as
    select FORMAT_OUT (sex) from t;

commit;

export database filename badimp into exp;
drop database filename badimp;
import database from exp filename badimp;
```

This restriction will be lifted in a future release of Oracle Rdb. Currently the workaround is to save the view definitions and reapply them after the IMPORT completes.

This restriction does not apply to external functions, created using the CREATE FUNCTION statement, as these database objects are defined before tables and views.

5.0.13 AIJSERVER Privileges

For security reasons, the AIJSERVER account ("RDMAIJSERVER") is created with only NETMBX and TMPMBX privileges. These privileges are sufficient to start Hot Standby, in most cases.

However, for production Hot Standby systems, these privileges are not adequate to ensure continued replication in all environments and workload situations. Therefore, Oracle recommends that the DBA provide the following additional privileges for the AIJSERVER account:

- ALTPRI

  This privilege allows the AIJSERVER to adjust its own priority to ensure adequate quorum (CPU utilization) to prompt message processing.

- PSWAPM

  This privilege allows the AIJSERVER to enable and disable process swapping, also necessary to ensure prompt message processing.

- SETPRV

  This privilege allows the AIJSERVER to temporarily set any additional privileges it may need to access the standby database or its server processes.

- SYSPRV

  This privilege allows the AIJSERVER to access the standby database rootfile, if necessary.

- WORLD

  This privilege allows the AIJSERVER to more accurately detect standby database server process failure and handle network failure more reliably.

5.0.14 Lock Remastering and Hot Standby

When using the Hot Standby feature, Oracle recommends that the VMS distributed lock manager resource tree be mastered on the standby node where Hot Standby is started. This can be using any of the following methods:

- Disable dynamic lock remastering. This can be done dynamically by setting the SYSGEN parameter PE1 to the value 1.

  When using this option, be sure that Hot Standby is started on the node where the standby database is first opened.

- Increasing the LOCKDIRWT value for the LRS node higher than any other node in the same cluster. However, this is not a dynamic SYSGEN parameter, and a node re-boot is required.

Failure to prevent dynamic lock remastering may cause severe performance degradation for the standby database, which ultimately may be reflected by decreased master database transaction throughput.

5.0.15 RDB_SETUP Privilege Error

Rdb Web Agent V3.0 exposes a privilege problem with Rdb V7.0 and later. This will be fixed in the next Rdb7 release.

The RDB_SETUP function fails with %RDB-E-NO_PRIV, privilege denied by database facility.

It appears that the only workaround is to give users DBADM privilege. Oracle Corporation does not recommend giving users the DBADM privilege.

5.0.16 Starting Hot Standby on Restored Standby Database May Corrupt Database

If a standby database is modified outside of Hot Standby, then backed up and restored, Hot Standby will appear to start up successfully but will corrupt the standby database. A subsequent query of the database will return unpredictable results, possibly in a bugcheck in DIOFETCH$FETCH_ONE_LINE. When the standby database is restored from a backup of itself, the database is marked as unmodified. Therefore, Hot Standby cannot tell whether the database had been modified before the backup was taken.

WORKAROUND: None.

5.0.17 Restriction on Compound Statement Nesting Levels

The use of multiple nesting levels of compound statements such as CASE or IF-THEN-ELSE within multistatement procedures can result in excessive memory usage during the compile of the procedure. Virtual memory problems have been reported with 10 or 11 levels of nesting. The following example shows an outline of the type of nesting that can lead to this problem.

```
CREATE MODULE MY_MOD LANGUAGE SQL
PROCEDURE MY PROCEDURE
    ( PARAMETERS .....);

BEGIN
  DECLARE ....;

SET :VARS = 0;
```

```
SELECT .....;
GET DIAGNOSTICS EXCEPTION 1 :FLAG = RETURNED_SQLCODE;
CASE :FLAG                      ! Case #1
       WHEN 100 THEN SET ...;
       WHEN -811 THEN SET ...;
       WHEN 0 THEN
         SET ...; SELECT ...;
         GET DIAGNOSTICS EXCEPTION 1 :FLAG = RETURNED_SQLCODE;
         CASE :FLAG            ! Case #2
            WHEN 0 THEN SET ...;
            WHEN -811 THEN SET ...;
            WHEN 100 THEN
               UPDATE...; SET ...;
               GET DIAGNOSTICS EXCEPTION 1 :FLAG = RETURNED_SQLCODE;
               IF :FLAG= 100 THEN SET ...;              ! #1
               ELSE
                IF :FLAG < 0 THEN SET...;               ! #2
               ELSE
                   DELETE  ...
                   GET DIAGNOSTICS EXCEPTION 1 :FLAG = RETURNED_SQLCODE;
                   IF :FLAG= 100 THEN SET...;           ! #3
                      SET ...;
                   ELSE
                    IF :FLAG < 0 THEN SET...;           ! #4
                    ELSE
                     IF IN_CHAR_PARAM = 'S' THEN        ! #5
                      UPDATE ...
                      GET DIAGNOSTICS EXCEPTION 1 :FLAG = RETURNED_SQLCODE;
                        IF :FLAG= 100 THEN SET ...;     ! #6
                        ELSE
                         IF :FLAG < 0 THEN SET...;      ! #7
                         END IF;                        ! #7
                       END IF;                         ! #6
                     END IF;                           ! #5

                     IF :FLAG = 0 THEN                 ! #5
                       UPDATE ...
                       GET DIAGNOSTICS EXCEPTION 1 :FLAG = RETURNED_SQLCODE;
                       IF :FLAG= 100 THEN SET ...;      ! #6
                       ELSE
                          IF :FLAG < 0 THEN SET ...;    ! #7
                          ELSE
                             DELETE ...
                             GET DIAGNOSTICS EXCEPTION 1 :FLAG = RETURNED_SQLCODE:
                             IF :FLAG= 100 THEN SET ...;        ! #8
                             ELSE
                               IF :FLAG < 0 THEN SET ...;       ! #9
                               ELSE
                                 DELETE ...;
                                 GET DIAGNOSTICS EXCEPTION 1 :FLAG = RETURNED_SQLCODE;
                                 IF :FLAG= 100 THEN SET ...;    ! #10
                                    SET ...;
                                 ELSE
                                    IF :FLAG < 0 THEN SET ...;  ! #11
                                    END IF; (11 end if's  for #11 - #1)
            ELSE SET ...;
            END CASE;               ! Case #2
       ELSE SET ...;
       END CASE;                    ! Case #1
END;
END MODULE;
```

Workaround: Reduce the complexity of the multistatement procedure. Use fewer levels of compound statement nesting by breaking the multistatement procedure into smaller procedures or by using the CALL statement to execute nested stored procedures.

5.0.18 Back Up All AIJ Journals Before Performing a Hot Standby Switchover Operation

Prior to performing a proper Hot Standby switchover operation from the old master database to the new master database (old standby database), be sure to back up ALL AIJ journals.

If you do not back up the AIJ journals on the old master database prior to switchover, they will be initialized by the Hot Standby startup operation, and you will not have a backup of those AIJ journals.

Failure to back up these journals may place your new master database at risk of not being able to be recovered, requiring another fail-over in the event of system failure.

5.0.19 Concurrent DDL and Read-Only Transaction on the Same Table Not Compatible

It is possible that a read-only transaction could generate a bugcheck at DIOBND$FETCH_AIP_ENT + 1C4 if there is an active, uncommitted transaction that is making metadata changes to the same table. Analysis shows that the snapshot transaction is picking up stale metadata information. Depending on what metatdata modifications are taking place, it is possible for metadata information to be removed from the system tables but still exist in the snapshot file. When the read-only transaction tries to use that information, it no longer exists and causes a bugcheck.

The following example shows the actions of the two transactions:

```
A:                              B:
attach
set transaction read write
                                attach
                                set transaction read only
drop index emp_last_name
                                select * from employees
                                ...bugcheck...
```

The only workaround is to avoid running the two transactions together.

5.0.20 Oracle Rdb and the SRM_CHECK Tool

The Alpha Architecture Reference Manual, Third Edition (AARM) describes strict rules for using interlocked memory instructions. The Compaq Alpha 21264 (EV6) processor and all future Alpha processors are more stringent than their predecessors in their requirement that these rules be followed. As a result, code that has worked in the past despite noncompliance may now fail when executed on systems featuring the new 21264 processor.

Oracle Rdb Release 7.0.3 supports the Compaq Alpha 21264 (EV6) processor. Oracle has performed extensive testing and analysis of the Rdb code to ensure that it is compliant with the rules for using interlocked memory instructions.

However, customers using the Compaq supplied SRM_CHECK tool may find that several of the Oracle Rdb images cause the tool to report potential alpha architecture violations. Although SRM_CHECK can normally identify a code section in an image by the section's attributes, it is possible for OpenVMS images to contain data sections with those same attributes. As a result, SRM_CHECK may scan data as if it were code, and occasionally, a block of data may look like a noncompliant code sequence. This is the case with the Oracle Rdb supplied images. There is no actual instruction stream violation.

However, customers must use the SRM_CHECK tool on their own application executable image files. It is possible that applications linked with very old version of Oracle Rdb (versions prior to Oracle Rdb Release 6.0-05) could have included illegal interlocked memory instruction sequences produced by very old versions of compilers. This code was included in the Oracle Rdb object library files for some very old versions of Oracle Rdb.

If errant instruction sequences are detected in the objects supplied by the Oracle Rdb object libraries, the correct action is to relink the application with a more-current version of Oracle Rdb.

Additional information about the Compaq Alpha 21264 (EV6) processor interlocked memory instructions issues is available at:

http://www.openvms.digital.com/openvms/21264_considerations.html

5.0.21 Oracle RMU Checksum_Verification Qualifier

The Oracle Rdb RMU BACKUP database backup command includes a Checksum_ Verification qualifier.

Specifying Checksum_Verification requests that the RMU Backup command verify the checksum stored on each database page before it is backed up, thereby providing end-to-end error detection on the database I/O.

The Checksum_Verification qualifier uses additional CPU resources but can provide an extra measure of confidence in the quality of the data backed up. Use of the Checksum_Verification qualifier offers an additional level of data security and use of the Checksum_Verification qualifier permits Oracle RMU to detect the possibility that the data it is reading from these disks has only been partially updated.

Note, however, that if you specify the Nochecksum_Verification qualifier, and undetected corruptions exist in your database, the corruptions are included in your backup file and restored when you restore the backup file. Such a corruption might be difficult to recover from, especially if it is not detected until weeks or months after the restore operation is performed.

Oracle Corporation recommends that you use the Checksum_Verification qualifier with all database backup operations because of the improved data integrity this qualifier provides.

Unfortunately, due to an oversight, for versions of Oracle Rdb prior to Version 8.0, the default for online backups is the Nochecksum_Verification qualifier. When you do not specify the Checksum_Verification qualifie on all of your RMU database backup commands.

5.0.22 Do Not Use HYPERSORT with RMU/OPTIMIZE/AFTER_JOURNAL (Alpha)

OpenVMS Alpha V7.1 introduced the high-performance Sort/Merge utility (also known as HYPERSORT). This utility takes advantage of the Alpha architecture to provide better performance for most sort and merge operations.

The high-performance Sort/Merge utility supports a subset of the SOR routines. Unfortunately, the high-performance Sort/Merge utility does not support several of the interfaces used by the RMU/OPTIMIZE/AFTER_JOURNAL command. In addition, the high-performance Sort/Merge utility reports no error or warning when being called with the unsupported options used by the RMU/OPTIMIZE /AFTER_JOURNAL command.

For this reason, the use of the high-performance Sort/Merge utility is not supported for the RMU/OPTIMIZE/AFTER_JOURNAL command. Do not define the logical name SORTSHR to reference HYPERSORT.EXE.

5.0.23 Restriction on Using /NOONLINE with Hot Standby

When a user process is performing a read-only transaction on a standby database, an attempt to start replication on the standby database with the /NOONLINE qualifier will fail with the following error, and the database will be closed clusterwide:

```
%RDMS-F-OPERCLOSE, database operator requested database shutdown
```

In a previous release, the following error was returned and the process doing the read-only transaction was not affected:

```
%RDMS-F-STBYDBINUSE, standby database cannot be exclusively accessed for
replication
```

As a workaround, if exclusive access is necessary to the standby database, terminate any user processes before starting replication with the /NOONLINE qualifier.

This restriction is due to another bug fix and will be lifted in a future release of Oracle Rdb.

5.0.24 SELECT Query May Bugcheck with PSII2SCANGETNEXTBBCDUPLICATE Error

Bug 683916

A bugcheck could occur when a ranked B-tree index is used in a query after a database has been upgraded to Release 7.0.1.3. This is a result of index corruption that was introduced in previous versions of Oracle Rdb7. This corruption has been fixed and indexes created using Release 7.0.1.3 will not be impacted.

As a workaround, delete the affected index and re-create it under Oracle Rdb7 Release 7.0.1.3 or later.

5.0.25 DBAPack for Windows 3.1 is Deprecated

Oracle Enterprise Manager DBAPack will no longer be supported for use on Windows 3.1.

5.0.26 Determining Mode for SQL Non-Stored Procedures

Bug 506464.

Although stored procedures allow parameters to be defined with the modes IN, OUT, and INOUT, there is no similar mechanism provided for SQL module language or SQL precompiled procedures. However, SQL still associates a mode with a parameter using the following rules:

Any parameter which is the target of an assignment is considered an OUT parameter. Assignments consist of the following:

- The parameter is assigned a value with the SET or GET DIAGNOSTICS statement. For example:

```
set :p1 = 0;
get diagnostics :p2 = TRANSACTION_ACTIVE;
```

- The parameter is assigned a value with the INTO clause of an INSERT, UPDATE, or SELECT statement. For example:

```
insert into T (col1, col2)
    values (...)
    returning dbkey into :p1;

update accounts
    set account_balance = account_balance + :amount
    where account_number = :p1
    returning account_balance
    into :current_balance;

select last_name
    into :p1
    from employees
    where employee_id = '00164';
```

- The parameter is passed on a CALL statement as an OUT or INOUT argument. For example:

```
begin
call GET_CURRENT_BALANCE (:p1);
end;
```

Any parameter that is the source for a query is considered an IN parameter. Query references include:

- The parameter appears in the SELECT list, WHERE or HAVING clauses of a SELECT, or DELETE statement. For example:

```
select :p1 || last_name, count(*)
    from T
    where last_name like 'Smith%'
    group by last_name
    having count(*) > :p2;

delete from T
    where posting_date < :p1;
```

- The parameter appears on the right side of the assignment in a SET statement or SET clause of an UPDATE statement. For example:

```
set :p1 = (select avg(salary)
            from T
            where department = :p2);
update T
    set col1 = :p1
    where ...;
```

- The parameter is used to provide a value to a column in an INSERT statement. For example:

```
insert into T (col1, col2)
    values (:p1, :p2);
```

- The parameter is referenced by an expression in a TRACE, CASE, IF/ELSEIF, WHILE statement, or by the DEFAULT clause of a variable declaration. For example:

```
begin
declare :v integer default :p1;
DO_LOOP:
while :p2 > :p1
loop
    if :p1 is null then
        leave DO_LOOP;
    end if;
    set :p2 = :p2 + 1;
    ...;
    trace 'Loop at ', :p2;
end loop;
end;
```

- The parameter is passed on a CALL statement as an INOUT or IN argument. For example:

```
begin
call SET_LINE_SPEED (:p1);
end;
```

SQL only copies values from the client (application parameters) to the procedure running in the database server if it is marked as either an IN or INOUT parameter. SQL only returns values from the server to the client application parameter variables if the parameter is an OUT or INOUT parameter.

If a parameter is considered an OUT only parameter, then it must be assigned a value within the procedure, otherwise the result returned to the application is considered undefined. This could occur if the parameter is used within a conditional statement such as CASE or IF/ELSEIF. In the following example, the value returned by :p2 would be undefined if :p1 were negative or zero:

```
begin
if :p1 > 0 then
    set :p2 = (select count(*)
                from T
                where col1 = :p1);
end if;
end;
```

It is the responsibility of the application programmer to ensure that the parameter is correctly assigned values within the procedure. A workaround is to either explicitly initialize the OUT parameter, or make it an INOUT parameter. For example:

```
begin
if :p1 > 0 then
    set :p2 = (select count(*)
                from T
                where col1 = :p1);
elseif :p2 is null then
    begin
    end;
end if;
end;
```

The empty statement will include a reference to the parameter to make it an IN parameter as well as an OUT parameter.

5.0.27 DROP TABLE CASCADE Results in %RDB-E-NO_META_UPDATE Error

An error could result when a DROP TABLE CASCADE statement is issued. This occurs when the following conditions apply:

- A table is created with an index defined on the table.
- A storage map is created with a placement via index.
- The storage map is a vertical record partition storage map with two or more STORE COLUMNS clauses.

The error message given is %RDB-E-NO_META_UPDATE, metadata update failed.

The following example shows a table, index, and storage map definition followed by a DROP TABLE CASCADE statement and the resulting error message:

```
SQL> CREATE TABLE VRP_TABLE ( ID INT, ID2 INT);
SQL> COMMIT;
SQL> CREATE UNIQUE INDEX VRP_IDX ON VRP_TABLE (ID)
SQL> STORE IN EMPIDS_LOW;
SQL> COMMIT;
SQL> CREATE STORAGE MAP VRP_MAP
cont> FOR VRP_TABLE
cont> PLACEMENT VIA INDEX VRP_IDX
cont> ENABLE COMPRESSION
cont> STORE COLUMNS (ID)
cont> IN EMPIDS_LOW
cont> STORE COLUMNS (ID2)
cont> IN EMPIDS_MID;
SQL> COMMIT;
SQL>
SQL> DROP TABLE VRP_TABLE CASCADE;
SQL> -- Index VRP_IDX is also being dropped.
%RDB-E-NO_META_UPDATE, metadata update failed
-RDMS-E-WISH_LIST, feature not implemented yet
-RDMS-E-VRPINVALID, invalid operation for storage map "VRP_MAP"
```

The workaround to this problem is to first delete the storage map, and then delete the table using the CASCADE option. The following example shows the workaround. The SHOW statement indicates that the table, index, and storage map were deleted:

```
SQL> DROP STORAGE MAP VRP_MAP;
SQL> DROP TABLE VRP_TABLE CASCADE;
SQL> -- Index VRP_IDX is also being dropped.
SQL> COMMIT;
SQL> SHOW TABLE VRP_TABLE
No tables found
SQL> SHOW INDEX VRP_IDX
No indexes found
SQL> SHOW STORAGE MAP VRP_MAP
No Storage Maps Found
```

This problem will be corrected in a future version of Oracle Rdb.

5.0.28 Bugcheck Dump Files with Exceptions at COSI_CHF_SIGNAL

In certain situations, Oracle Rdb bugcheck dump files will indicate an exception at COSI_CHF_SIGNAL. This location is, however, not the address of the actual exception. The actual exception occurred at the previous call frame on the stack (the one listed as the next "Saved PC" after the exception).

For example, consider the following bugcheck file stack information:

```
$ SEARCH RDSBUGCHK.DMP "EXCEPTION","SAVED PC","-F-","-E-"

***** Exception at 00EFA828 : COSI_CHF_SIGNAL + 00000140
%COSI-F-BUGCHECK, internal consistency failure
Saved PC = 00C386F0 : PSIINDEX2JOINSCR + 00000318
Saved PC = 00C0BE6C : PSII2BALANCE + 0000105C
Saved PC = 00C0F4D4 : PSII2INSERTT + 000005CC
Saved PC = 00C10640 : PSII2INSERTTREE + 000001A0
     .
     .
     .
```

In this example, the exception actually occurred at PSIINDEX2JOINSCR offset 00000318. If you have a bugcheck dump with an exception at COSI_CHF_ SIGNAL, it is important to note the next “Saved PC” because it will be needed when working with Oracle Rdb Support Services.

5.0.29 Interruptions Possible when Using Multistatement or Stored Procedures

Long running multistatement or stored procedures can cause other users in the database to be interrupted by holding resources needed by those other users. Some resources obtained by the execution of a multistatement or stored procedure will not be released until the multistatement or stored procedure finishes. This problem can be encountered even if the statement contains COMMIT or ROLLBACK statements.

The following example demonstrates the problem. The first session enters an endless loop; the second session attempts to backup the database, but it is permanently interrupted:

Session 1

```
SQL> ATTACH 'FILE MF_PERSONNEL';
SQL> CREATE FUNCTION LIB$WAIT (IN REAL BY REFERENCE)
cont> RETURNS INT;
cont> EXTERNAL NAME LIB$WAIT
cont> LOCATION 'SYS$SHARE:LIBRTL.EXE'
cont> LANGUAGE GENERAL
cont> GENERAL PARAMETER STYLE
cont> VARIANT;
SQL> COMMIT;
SQL> EXIT;

$ SQL
SQL> ATTACH 'FILE MF_PERSONNEL';
SQL> BEGIN
cont> DECLARE :LAST_NAME LAST_NAME_DOM;
cont> DECLARE :WAIT_STATUS INTEGER;
cont> LOOP
cont> SELECT LAST_NAME INTO :LAST_NAME
cont>   FROM EMPLOYEES WHERE EMPLOYEE_ID = '00164';
cont> ROLLBACK;
cont> SET :WAIT_STATUS = LIB$WAIT (5.0);
cont> SET TRANSACTION READ ONLY;
cont> END LOOP;
cont> END;
```

Session 2

```
$ RMU/BACKUP/LOG/ONLINE MF_PERSONNEL MF_PERSONNEL
```

From a third session we can see that the backup process is waiting for a lock held in the first session:

```
$ RMU/SHOW LOCKS /MODE=BLOCKING MF_PERSONNEL
================================================================================
SHOW LOCKS/BLOCKING Information
================================================================================

--------------------------------------------------------------------------------
Resource: nowait signal

          ProcessID Process Name        Lock ID   System ID Requested Granted
          --------- ---------------     --------- --------- --------- -------
Waiting:  20204383  RMU BACKUP.....     5600A476  00010001  CW        NL
Blocker:  2020437B  SQL............     3B00A35C  00010001  PR        PR
$
```

There is no workaround for this restriction. When the multistatement or stored procedure finishes execution, the resources needed by other processes will be released.

5.0.30 Row Cache Not Allowed on Standby Database While Hot Standby Replication Is Active

The row cache feature may not be active on a Hot Standby database while replication is taking place. The Hot Standby feature will not start if row cache is active on the standby database.

This restriction exists because rows in the row cache are accessed using logical dbkeys. However, information transferred to the Hot Standby database from the after-image journal facility only contains physical dbkeys. Because there is no way to maintain rows in the cache using 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.

The row cache feature should be identically configured on the master and standby databases in the event failover occurs, but the row cache feature must not be activated on the standby database until it becomes the master.

A new command qualifier, ROW_CACHE=DISABLED, has been added to the RMU/OPEN command to disable the row cache feature on the standby database. To open the Hot Standby database prior to starting replication, use the ROW_CACHE=DISABLED qualifier on the RMU/OPEN command.

5.0.31 Hot Standby Replication Waits when Starting if Read-Only Transactions Running

Hot Standby replication will wait to start if there are read-only (snapshot) transactions running on the standby database. The log roll-forward server (LRS) will wait until the read-only transactions commit, and then replication will continue.

This is an existing restriction of the Hot Standby software. This release note is intended to complement the Hot Standby documentation.

5.0.32 Error when Using the SYS$LIBRARY:SQL_FUNCTIONS70.SQL Oracle Functions Script

If your programming environment is not set up correctly, you may encounter problems running the SYS$LIBRARY:SQL_FUNCTIONS70.SQL script used to set up the Oracle7 functions being supplied with Oracle Rdb.

The following example shows the error:

```
%RDB-E-EXTFUN_FAIL, external routine failed to compile or execute successfully
-RDMS-E-INVRTNUSE, routine RDB$ORACLE_SQLFUNC_INTRO can not be used, image
"SQL$FUNCTIONS" not activated
-RDMS-I-TEXT, Error activating image
DISK:[DIR]SQL$FUNCTIONS.;, File not found
```

To resolve this problem, use the @SYS$LIBRARY:RDB$SETVER to set up the appropriate logical names. This will be necessary for programs that use the functions as well.

In a standard environment, use the command shown in the following example:

```
$ @SYS$LIBRARY:RDB$SETVER S
```

In a multiversion environment, use the command shown in the following example:

```
$ @SYS$LIBRARY:RDB$SETVER 70
```

5.0.33 DEC C and Use of the /STANDARD Switch

Bug 394451

The SQL$PRE compiler examines the system to know which dialect of C to generate. That default can be overwritten by using the /CC=[DECC/VAXC] switch. The /STANDARD switch should not be used to choose the dialect of C.

Support for DEC C was added to the product with V6.0 and this note is meant to clarify that support, not to indicate a change. It is possible to use /STANDARD=RELAXED_ANSI89 or /STANDARD=VAXC correctly, but this is not recommended.

The following example shows both the right and wrong way to compile an Oracle Rdb SQL program. Assume a symbol SQL$PRE has been defined, and DEC C is the default C compiler on the system:

```
$ SQL$PRE/CC  ! This is correct.
$ SQL$PRE/CC=DECC ! This is correct.
$ SQL$PRE/CC=VAXC ! This is correct.

$ SQL$PRE/CC/STANDARD=VAXC   ! This is incorrect.
```

Notice that the /STANDARD switch has other options in addition to RELAXED_ANSI89 and VAX C. Those are also not supported.

5.0.34 Excessive Process Page Faults and Other Performance Considerations During Oracle Rdb Sorts

Excessive hard or soft page faulting can be a limiting factor of process performance. Sometimes this page faulting occurs during Oracle Rdb sort operations. This note describes how page faulting can occur and some ways to help control, or at least understand, it.

One factor contributing to Oracle Rdb process page faulting is sorting operations. Common causes of sorts include the SQL GROUP BY, ORDER BY, UNION, and DISTINCT clauses specified for query and index creation operations. Defining the logical name RDMS$DEBUG_FLAGS to "RS" can help determine when Oracle Rdb sort operations are occurring and to display the sort keys and statistics.

Oracle Rdb includes its own copy of the OpenVMS SORT32 code within the Oracle Rdb images and does not generally call the routines in the OpenVMS run-time library. A copy of the SORT32 code is used to provide stability between versions of Oracle Rdb and OpenVMS and because Oracle Rdb calls the sort routines from executive processor mode which is difficult to do using the SORT32 sharable image. Database import and RMU load operations call the OpenVMS sort run-time library.

At the beginning of a sort operation, the sort code allocates some memory for working space. The sort code uses this space for buffers, in-memory copies of the data, and sorting trees.

Sort code does not directly consider the process quotas or parameters when allocating memory. The effects of WSQUOTA and WSEXTENT are indirect. At the beginning of each sort operation, the sort code attempts to adjust the process' working set to the maximum possible size using the $ADJWSL system service specifying a requested working set limit of %X7FFFFFFF pages (the maximum possible). Sort code then uses a value of 75% of the returned working set for virtual memory scratch space. The scratch space is then initialized and the sort begins.

The initialization of the scratch space generally causes page faults to access the pages newly added to the working set. Pages that were in the working set already may be faulted out as new pages are faulted in. Once the sort operation completes, the pages that may have been faulted out of the working set are likely to be faulted back into the working set.

When a process' working set is limited by the working set quota (WSQUOTA) parameter and the working set extent (WSEXTENT) parameter is a much larger value, the first call to the sort routines can cause many page faults as the working set grows. Using a value of WSEXTENT that is closer to WSQUOTA can help reduce the impact of this case.

With some OpenVMS versions, AUTOGEN sets the SYSGEN parameter PQL_MWSEXTENT equal to the WSMAX parameter. This means that all processes on the system end up with WSEXTENT the same as WSMAX. Because WSMAX might be quite high, sorting might result in excessive page faulting. You may want to explicitly set PQL_MWSEXTENT to a lower value if this is the case on your system.

Sort work files are another factor to consider when tuning Oracle Rdb sort operations. When the operation cannot be done in available memory, sort code will use temporary disk files to hold the data as it is being sorted. The *Oracle Rdb Guide to Performance and Tuning* contains more detailed information about sort work files.

The logical name RDMS$BIND_SORT_WORKFILES specifies how many work files sort code is to use if work files are required. The default is 2, and the maximum number is 10. The work files can be individually controlled by the SORTWORKn logical names (where n is from 0 through 9). You can increase the efficiency of sort operations by assigning the location of the temporary sort work files to different disks. These assignments are made by using up to 10 logical names, SORTWORK0 through SORTWORK9.

Normally, sort code places work files in the user's SYS$SCRATCH directory. By default, SYS$SCRATCH is the same device and directory as the SYS$LOGIN location. Spreading the I/O load over many disks improves efficiency as well as performance by taking advantage of the system resources and helps prevent disk I/O bottlenecks. Specifying that a user's work files will reside on separate disks permits overlap of the sort read/write cycle. You may also encounter cases where insufficient space exists on the SYS$SCRATCH disk device, such as when Oracle Rdb builds indexes for a very large table. Using the SORTWORK0 through SORTWORK9 logical names can help you avoid this problem.

Note that sort code uses the work files for different sorted runs, and then merges the sorted runs into larger groups. If the source data is mostly sorted, then not every sort work file may need to be accessed. This is a possible source of confusion because even with 10 sort work files, it is possible to exceed the capacity of the first sort file, and the sort operation will fail never having accessed the remaining 9 sort work files.

Note that the logical names RDMS$BIND_WORK_VM and RDMS$BIND_WORK_FILE do not affect or control the operation of sort. These logical names are used to control other temporary space allocations within Oracle Rdb.

5.0.35 Performance Monitor Column Mislabeled

The File IO Overview statistics screen, in the Rdb Performance Monitor, contains a column labeled Pages Checked. The column should be labeled Pages Discarded to correctly reflect the statistic displayed.

5.0.36 Restriction Using Backup Files Created Later than Oracle Rdb7 Release 7.0.1

Bug 521583

Backup files created using Oracle Rdb7 releases later than 7.0.1 cannot be restored using Oracle Rdb7 Release 7.0.1. To fix a problem in a previous release, some internal backup file data structures were changed. These changes are not backward compatible with Oracle Rdb7 Release 7.0.1.

If you restore the database using such a backup file, then any attempt to access the restored database may result in unpredictable behavior, even though a verify operation may indicate no problems.

There is no workaround to this problem. For this reason, Oracle Corporation strongly recommends performing a full and complete backup both before and after the upgrade from Release 7.0.1 to later releases of Oracle Rdb7.

5.0.37 RMU Backup Operations and Tape Drive Types

When using more than one tape drive for an RMU backup operation, all the tape drives must be of the same type. For example, all the tape drives must be either TA90s or TZ87s or TK50s. Using different tape drive types (one TK50 and one TA90) for a single database backup operation may make database restoration difficult or impossible.

Oracle RMU attempts to prevent 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 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 using a TA90 and a TA90E.

Oracle recommends that, on a regular basis, you test your backup and recovery procedures and environment using a test system. You should restore the databases and then recover them using AIJs to simulate failure recovery of the production system.

Consult the *Oracle Rdb Guide to Database Maintenance*, the *Oracle Rdb Guide to Database Design and Definition*, and the *Oracle RMU Reference Manual* for additional information about Oracle Rdb backup and restore operations.

5.0.38 Use of Oracle Rdb from Shared Images

Bug 470946

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 Oracle Rdb's images have not had a chance to do their own initialization.

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

- Do not make Oracle Rdb calls from the initialization routines of shared images.
- Link in such a way that the RDBSHR.EXE image initializes first. This can be done by placing the reference to RDBSHR.EXE and any other Oracle Rdb shared images last in the linker options file.

5.0.39 Restriction Added for CREATE STORAGE MAP on Table with Data

Oracle Rdb7 added support that allows a storage map to be added to an existing table which contains data. The restrictions listed for Oracle Rdb7 were:

- The storage map 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. It can only contain one area and cannot be vertically partitioned. This new map simply describes the mapping as it exists by default for the table.

This release of Rdb7 adds the additional restriction that the storage map may not include a WITH LIMIT clause for the storage area. The following example shows the reported error:

```
SQL> CREATE TABLE MAP_TEST1 (A INTEGER, B CHAR(10));
SQL> CREATE INDEX MAP_TEST1_INDEX ON MAP_TEST1 (A);
SQL> INSERT INTO MAP_TEST1 (A, B) VALUES (3, 'Third');
1 row inserted
SQL> CREATE STORAGE MAP MAP_TEST1_MAP FOR MAP_TEST1
cont> STORE USING (A) IN RDB$SYSTEM
cont>     WITH LIMIT OF (10);  -- can't use WITH LIMIT clause
%RDB-E-NO_META_UPDATE, metadata update failed
-RDMS-F-RELNOTEMPTY, table "MAP_TEST1" has data in it
-RDMS-E-NOCMPLXMAP, can not use complex map for non-empty table
```

5.0.40 ALTER DOMAIN...DROP DEFAULT Reports DEFVALUNS Error

Bug 456867

If a domain has a DEFAULT of CURRENT_USER, SESSION_USER, or SYSTEM_USER and attempts to delete that default, it may fail unexpectedly. The following example shows the error:

```
SQL> ATTACH 'FILENAME PERSONNEL';
SQL> CREATE DOMAIN ADDRESS_DATA2_DOM CHAR(31)
cont>  DEFAULT CURRENT_USER;
SQL> COMMIT;
SQL> ALTER DOMAIN ADDRESS_DATA2_DOM
cont> DROP DEFAULT;
%SQL-F-DEFVALUNS, Default values are not supported for the data type of
ADDRESS_DATA2_DOM
```

To work around this problem you must first alter the domain to have a default of NULL, as shown, and then use DROP DEFAULT:

```
SQL> ALTER DOMAIN ADDRESS_DATA2_DOM
cont> SET DEFAULT NULL;
SQL> ALTER DOMAIN ADDRESS_DATA2_DOM
cont> DROP DEFAULT;
SQL> COMMIT;
```

This problem will be corrected in a future release of Oracle Rdb.

5.0.41 Oracle Rdb7 Workload Collection Can Stop Hot Standby Replication

If you are replicating your Oracle Rdb7 database using the Oracle Hot Standby option, you must not use the workload collection option. By default, workload collection is disabled. However, if you enabled workload collection, you must disable it on the master database prior to performing a backup operation on that master database if it will be used to create the standby database for replication purposes. If you do not disable workload collection, it could write workload information to the standby database and prevent replication operations from occurring.

The workaround included at the end of this section describes how to disable workload collection on the master database and allow the Hot Standby software to propagate the change to the standby database automatically during replication operations.

Background Information

By default, workload collection and cardinality collection are automatically disabled when Hot Standby replication operations are occurring on the standby database. However, if replication stops (even for a brief network failure), Oracle Rdb7 potentially can start a read/write transaction on the standby database to write workload collection information. Then, because the standby database is no longer synchronized transactionally with the master database, replication operations cannot restart.

Note

The Oracle Rdb7 optimizer can update workload collection information in the RDB$WORKLOAD system table even though the standby database is opened exclusively for read-only queries. A read/write transaction is started during the disconnection from the standby database to flush the workload and cardinality statistics to the system tables.

If the standby database is modified, you receive the following messages when you try to restart Hot Standby replication operations:

```
%RDMS-F-DBMODIFIED, database has been modified; AIJ roll-forward not possible
%RMU-F-FATALRDB, Fatal error while accessing Oracle Rdb.
```

Workaround

To work around this problem, perform the following:

- On the master database, disable workload collection using the SQL clause WORKLOAD COLLECTION IS DISABLED on the ALTER DATABASE statement. For example:

  ```
  SQL> ALTER DATABASE FILE mf_personnel
  cont> WORKLOAD COLLECTION IS DISABLED;
  ```

  This change is propagated to the standby database automatically when you restore the standby database and restart replication operations. Note that, by default, the workload collection feature is disabled. You need to disable workload collection only if you previously enabled workload collection with the WORKLOAD COLLECTION IS ENABLED clause.

- On the standby database, include the Transaction_Mode qualifier on the RMU/Restore command when you restore the standby database. You should set this qualifier to read-only to prevent modifications to the standby database when replication operations are not active. The following example shows the Transaction_Mode qualifier used in a typical RMU/Restore command:

  ```
  $ RMU/RESTORE /TRANSACTION_MODE=READ_ONLY
                /NOCDD
                /NOLOG
                /ROOT=DISK1:[DIR]standby_personnel.rdb
                /AIJ_OPT=aij_opt.dat
                DISK1:[DIR]standby_personnel.rbf
  ```

If, in the future, you fail over processing to the standby database (so that the standby database becomes the master database), you can re-enable updates to the "new" master database. For example, to re-enable updates, use the SQL statement ALTER DATABASE and include the SET TRANSACTION MODES (ALL) clause. The following example shows this statement used on the new master database:

```
SQL> ALTER DATABASE FILE mf_personnel
cont> SET TRANSACTION MODES (ALL);
```

5.0.42 RMU Convert Command and System Tables

When the RMU Convert command converts a database from a previous version to Oracle Rdb V7.0 or higher, it sets the RDB$CREATED and RDB$LAST_ ALTERED columns to the timestamp of the convert operation.

The RDB$xxx_CREATOR columns are set to the current user name (which is space filled) of the converter. Here xxx represents the object name, such as in RDB$TRIGGER_CREATOR.

The RMU Convert command also creates the new index on RDB$TRANSFER_ RELATIONS if the database is transfer enabled.

5.0.43 Converting Single-File Databases

Because of a substantial increase in the database root file information for Release 7.0, you should ensure that you have adequate disk space before you use the RMU Convert command with single-file databases and Release 7.0 or higher.

The size of the database root file of any given database will increase a minimum of 13 blocks and a maximum of 597 blocks. The actual increase depends mostly on the maximum number of users specified for the database.

5.0.44 Restriction when Adding Storage Areas with Users Attached to Database

If you try to interactively add a new storage area where the page size is smaller than the smallest existing page size and the database has been manually opened or users are active, the add operation fails with the following error:

```
%RDB-F-SYS_REQUEST, error from system services request
-RDMS-F-FILACCERR, error opening database root DKA0:[RDB]TEST.RDB;1
-SYSTEM-W-ACCONFLICT, file access conflict
```

You can make this change only when no users are attached to the database and, if the database is set to OPEN IS MANUAL, the database is closed. Several internal Oracle Rdb data structures are based on the minimum page size, and these structures cannot be resized if users are attached to the database.

Furthermore, because this particular change is not recorded in the AIJ file, any recovery scenario will fail. Note also that if you use .aij files, you must backup the database and restart after-image journaling because this change invalidates the current AIJ recovery.

5.0.45 Restriction on Tape Usage for Digital UNIX V3.2

5.0.46 Support for Single-File Databases to be Dropped in a Future Release

Oracle Rdb currently supports both single-file and multifile databases on OpenVMS. However, single-file databases will not be supported in a future release of Oracle Rdb. At that time, Oracle Rdb will provide the means to easily convert single-file databases to multifile databases.

Oracle recommends that users with single-file databases perform the following actions:

- Use the Oracle RMU commands, such as Backup and Restore, to make copies, back up, or move single-file databases. Do not use operating system commands to copy, back up, or move databases.

- Create new databases as multifile databases even though single-file databases are supported in Oracle Rdb release 6.1 and release 7.0.

5.0.47 DECdtm Log Stalls

Resource managers using the DECdtm services can sometimes suddenly stop being able to commit transactions. If Oracle Rdb7 is installed and transactions are being run, an RMU Show command on the affected database will show transactions as being "stalled, waiting to commit".

Refer to the DECdtm documentation and release notes for information on symptoms, fixes, and workarounds for this problem. One workaround, for OpenVMS V5.5-x, is provided here.

On the affected node while the log stall is in progress, type the following command from a privileged account:

```
$ MCR LMCP SET NOTIMEZONE
```

This should force the log to restart.

This stall occurs only when a particular bit in a pointer field becomes set. To see the value of the pointer field, enter the following command from a privileged account (where <nodename> is the SCS node name of the node in question).

```
$ MCR LMCP DUMP/ACTIVE/NOFORM SYSTEM$<nodename>
```

This command displays output similar to the following:

```
Dump of transaction log SYS$COMMON:[SYSEXE]SYSTEM$<nodename>.LM$JOURNAL;1
End of file block 4002 / Allocated 4002
Log Version 1.0
Transaction log UID:   29551FC0-CBB7-11CC-8001-AA000400B7A5
Penultimate Checkpoint: 000013FD4479 0079
Last Checkpoint:        000013FDFC84 0084

Total of 2 transactions active, 0 prepared and 2 committed.
```

The stall will occur when bit 31 of the checkpoint address becomes set, as this excerpt from the previous example shows:

```
        Last Checkpoint:        000013FDFC84 0084
                                    ^
                                    |
```

When the number indicated in the example becomes 8, the log will stall. Check this number and observe how quickly it grows. When it is at 7FFF, frequently use the following command:

```
$ MCR LMCP SHOW LOG /CURRENT
```

If this command shows a stall in progress, use the workaround to restart the log.

See your Compaq Computer Corporation representative for information about patches to DECdtm.

5.0.48 Cannot Run Distributed Transactions on Systems with DECnet/OSI and OpenVMS Alpha Version 6.1 or OpenVMS VAX Version 6.0

If you have DECnet/OSI installed on a system with OpenVMS Alpha Version 6.1 or OpenVMS VAX Version 6.0, you cannot run Oracle Rdb7 operations that require the two-phase commit protocol. The two-phase commit protocol guarantees that if one operation in a distributed transaction cannot be completed, none of the operations is completed.

If you have DECnet/OSI installed on a system running OpenVMS VAX Version 6.1 or higher or OpenVMS Alpha Version 6.2 or higher, you can run Oracle Rdb operations that require the two-phase commit protocol.

For more information about the two-phase commit protocol, see the *Oracle Rdb Guide to Distributed Transactions*.

5.0.49 Multiblock Page Writes May Require Restore Operation

If a node fails while a multiblock page is being written to disk, the page in the disk becomes inconsistent and is detected immediately during failover. (Failover is the recovery of an application by restarting it on another computer.) The problem is rare and occurs because only single-block I/O operations are guaranteed by OpenVMS to be written atomically. This problem has never been reported by any customer and was detected only during stress tests in our labs.

Correct the page by an area-level restore operation. Database integrity is not compromised, but the affected area will not be available until the restore operation completes.

A future release of Oracle Rdb will provide a solution that guarantees multiblock atomic write operations. Cluster failovers will automatically cause the recovery of multiblock pages, and no manual intervention will be required.

5.0.50 Oracle Rdb7 Network Link Failure Does Not Allow DISCONNECT to Clean Up Transactions

If a program attaches to a database on a remote node and it loses the connection before the COMMIT statement is issued, there is nothing you can do except exit the program and start again.

The problem occurs when a program is connected to a remote database and updates the database, but then just before it commits, the network fails. When the commit executes, SQL shows, as it normally should, that the program has lost the link. Assume that the user waits for a minute or two, then tries the transaction again. The problem is that when the start transaction is issued for the second time, it fails because old information still exists about the previous failed transaction. This occurs even if the user issues a DISCONNECT statement (in Release 4.1 and earlier, a FINISH statement), which also fails with an RDB-E-IO_ERROR error message.

5.0.51 Replication Option Copy Processes Do Not Process Database Pages Ahead of an Application

When a group of copy processes initiated by the Replication Option (formerly Data Distributor) begins running after an application has begun modifying the database, the copy processes will catch up to the application and will not be able to process database pages that are logically ahead of the application in the RDB$CHANGES system table. The copy processes all align waiting for the same database page and do not move on until the application has released it. The performance of each copy process degrades because it is being paced by the application.

When a copy process completes updates to its respective remote database, it updates the RDB$TRANSFERS system table and then tries to delete any RDB$CHANGES rows not needed by any transfers. During this process, the RDB$CHANGES table cannot be updated by any application process, holding up any database updates until the deletion process is complete. The application stalls while waiting for the RDB$CHANGES table. The resulting contention

for RDB$CHANGES SPAM pages and data pages severely impacts performance throughput, requiring user intervention with normal processing.

This is a known restriction in Release 4.0 and higher. Oracle Rdb uses page locks as latches. These latches are held only for the duration of an action on the page and not to the end of transaction. The page locks also have blocking asynchronous system traps (ASTs) associated with them. Therefore, whenever a process requests a page lock, the process holding that page lock is sent a blocking AST (BLAST) by OpenVMS. The process that receives such a blocking AST queues the fact that the page lock should be released as soon as possible. However, the page lock cannot be released immediately.

Such work requests to release page locks are handled at verb commit time. An Oracle Rdb verb is an Oracle Rdb query that executes atomically, within a transaction. Therefore, verbs that require the scan of a large table, for example, can be quite long. An updating application does not release page locks until its verb has completed.

The reasons for holding on to the page locks until the end of the verb are fundamental to the database management system.

5.0.52 SQL Does Not Display Storage Map Definition After Cascading Delete of Storage Area

When you delete a storage area using the CASCADE keyword and that storage area is not the only area to which the storage map refers, the SHOW STORAGE MAP statement no longer shows the placement definition for that storage map.

The following example demonstrates this restriction:

```
SQL> SHOW STORAGE MAP DEGREES_MAP1
     DEGREES_MAP1
 For Table:             DEGREES1
 Compression is:        ENABLED
 Partitioning is:       NOT UPDATABLE
 Store clause:          STORE USING (EMPLOYEE_ID)
            IN DEG_AREA WITH LIMIT OF ('00250')
             OTHERWISE IN DEG_AREA2

SQL> DISCONNECT DEFAULT;
SQL> -- Drop the storage area, using the CASCADE keyword.
SQL> ALTER DATABASE FILENAME MF_PERSONNEL
cont> DROP STORAGE AREA DEG_AREA CASCADE;
SQL> --
SQL> -- Display the storage map definition.
SQL> ATTACH 'FILENAME MF_PERSONNEL';
SQL> SHOW STORAGE MAP DEGREES_MAP1
     DEGREES_MAP1
 For Table:             DEGREES1
 Compression is:        ENABLED
 Partitioning is:       NOT UPDATABLE

SQL>
```

The other storage area, DEG_AREA2, still exists, even though the SHOW STORAGE MAP statement does not display it.

A workaround is to use the RMU Extract command with the Items=Storage_Map qualifier to see the mapping.

5.0.53 ARITH_EXCEPT or Incorrect Results Using LIKE IGNORE CASE

When you use LIKE . . . IGNORE CASE, programs linked under Oracle Rdb Release 4.2 and Release 5.1, but run under higher versions of Oracle Rdb, may result in incorrect results or %RDB-E-ARITH_EXCEPT exceptions.

To work around the problem, avoid using IGNORE CASE with LIKE, or recompile and relink under a higher version (Release 6.0 or higher.)

5.0.54 Different Methods of Limiting Returned Rows from Queries

You can establish the query governor for rows returned from a query by using the SQL SET QUERY LIMIT statement, a logical name, or a configuration parameter. This note describes the differences between the mechanisms.

- If you define the RDMS$BIND_QG_REC_LIMIT logical name or RDB_BIND_QG_REC_LIMIT configuration parameter to a small value, the query will often fail with no rows returned. The following example demonstrates setting the limit to 10 rows and the resulting failure:

  ```
  $ DEFINE RDMS$BIND_QG_REC_LIMIT 10
  $ SQL$
  SQL> ATTACH 'FILENAME MF_PERSONNEL';
  SQL> SELECT EMPLOYEE_ID FROM EMPLOYEES;
  %RDB-F-EXQUOTA, Oracle Rdb runtime quota exceeded
  -RDMS-E-MAXRECLIM, query governor maximum limit of rows has been reached
  ```

  Interactive SQL must load its metadata cache for the table before it can process the SELECT statement. In this example, interactive SQL loads its metadata cache to allow it to check that the column EMPLOYEE_ID really exists for the table. The queries on the Oracle Rdb system tables RDB$RELATIONS and RDB$RELATION_FIELDS exceed the limit of rows.

  Oracle Rdb does not prepare the SELECT statement, let alone execute it. Raising the limit to a number less than 100 (the cardinality of EMPLOYEES) but more than the number of columns in EMPLOYEES (that is, the number of rows to read from the RDB$RELATION_FIELDS system table) is sufficient to read each column definition.

  To see an indication of the queries executed against the system tables, define the RDMS$DEBUG_FLAGS logical name or the RDB_DEBUG_FLAGS configuration parameter as S or B.

- If you set the row limit using the SQL SET QUERY statement and run the same query, it returns the number of rows specified by the SQL SET QUERY statement before failing:

  ```
  SQL> ATTACH 'FILENAME MF_PERSONNEL';
  SQL> SET QUERY LIMIT ROWS 10;
  SQL> SELECT EMPLOYEE_ID FROM EMPLOYEES;
  EMPLOYEE_ID
  00164
  00165
     .
     .
     .
  00173
  %RDB-E-EXQUOTA, Oracle Rdb runtime quota exceeded
  -RDMS-E-MAXRECLIM, query governor maximum limit of rows has been reached
  ```

  The SET QUERY LIMIT specifies that only user queries be limited to 10 rows. Therefore, the queries used to load the metadata cache are not restricted in any way.

Like the SET QUERY LIMIT statement, the SQL precompiler and module processor command line qualifiers (QUERY_MAX_ROWS and SQLOPTIONS=QUERY_MAX_ROWS) only limit user queries.

Keep the differences in mind when limiting returned rows using the logical name RDMS$BIND_QG_REC_LIMIT or the configuration parameter RDB_BIND_QG_REC_LIMIT. They may limit more queries than are obvious. This is important when using 4GL tools, the SQL precompiler, the SQL module processor, and other interfaces that read the Oracle Rdb system tables as part of query processing.

5.0.55 Suggestions for Optimal Usage of the SHARED DATA DEFINITION Clause for Parallel Index Creation

The CREATE INDEX process involves the following steps:

1. Process the metadata.
2. Lock the index name.

   Because new metadata (which includes the index name) is not written to disk until the end of the index process, Oracle Rdb must ensure index name uniqueness across the database during this time by taking a special lock on the provided index name.
3. Read the table for sorting by selected index columns and ordering.
4. Sort the key data.
5. Build the index (includes partitioning across storage areas).
6. Write new metadata to disk.

Step 6 is the point of conflict with other index definers because the system table and indexes are locked like any other updated table.

Multiple users can create indexes on the same table by using the RESERVING table_name FOR SHARED DATA DEFINITION clause of the SET TRANSACTION statement. For optimal usage of this capability, Oracle Rdb suggests the following guidelines:

- You should commit the transaction immediately after the CREATE INDEX statement so that locks on the table are released. This avoids lock conflicts with other index definers and improves overall concurrency.
- By assigning the location of the temporary sort work files SORTWORK0, SORTWORK1, . . . , SORTWORK9 to different disks for each parallel process that issues the SHARED DATA DEFINITION statement, you can increase the efficiency of sort operations. This minimizes any possible disk I/O bottlenecks and allows overlap of the SORT read/write cycle.
- If possible, enable global buffers and specify a buffer number large enough to hold a sufficient amount of table data. However, do not define global buffers larger than the available system physical memory. Global buffers allow sharing of database pages and thus result in disk I/O savings. That is, pages are read from disk by one of the processes and then shared by the other index definers for the same table, reducing the I/O load on the table.
- If global buffers are not used, ensure that enough local buffers exist to keep much of the index cached (use the RDM$BIND_BUFFERS logical name or RDB_BIND_BUFFERS configuration parameter or the NUMBER OF BUFFERS IS clause in SQL to change the number of buffers).

- To distribute the disk I/O load, place the storage areas for the indexes on separate disk drives. Note that using the same storage area for multiple indexes will result in contention during the index creation (Step 5) for SPAM pages.
- Consider placing the .ruj file for each parallel definer on its own disk or an infrequently used disk.
- Even though snapshot I/O should be minimal, consider disabling snapshots during parallel index creation.
- Refer to the *Oracle Rdb Guide to Performance and Tuning* to determine the appropriate working set values for each process to minimize excessive paging activity. In particular, avoid using working set parameters where the difference between WSQUOTA and WSEXTENT is large. The SORT utility uses the difference between these two values to allocate scratch virtual memory. A large difference (that is, the requested virtual memory grossly exceeds the available physical memory) may lead to excessive page faulting.
- The performance benefits of using SHARED DATA DEFINITION can best be observed when creating many indexes in parallel. The benefit is in the average elapsed time, not in CPU or I/O usage. For example, when two indexes are created in parallel using the SHARED DATA DEFINITION clause, the database must be attached twice, and the two attaches each use separate system resources.
- Using the SHARED DATA DEFINITION clause on a single-file database or for indexes defined in the RDB$SYSTEM storage area is not recommended.

The following table displays the elapsed time benefit when creating multiple indexes in parallel with the SHARED DATA DEFINITION clause. The table shows the elapsed time for 10 parallel process index creations (Index1, Index2, . . . Index10) and one process with 10 sequential index creations (All10). In this example, global buffers are enabled and the number of buffers is 500. The longest time for a parallel index creation is Index7 with an elapsed time of 00:02:34.64, compared to creating 10 indexes sequentially with an elapsed time of 00:03:26.66. The longest single parallel create index elapsed time is shorter than the elapsed time of creating all 10 of the indexes serially.

Index Create Job	Elapsed Time
Index1	00:02:22.50
Index2	00:01:57.94
Index3	00:02:06.27
Index4	00:01:34.53
Index5	00:01:51.96
Index6	00:01:27.57
Index7	00:02:34.64
Index8	00:01:40.56
Index9	00:01:34.43
Index10	00:01:47.44
All 10	00:03:26.66

5.0.56 Side Effect when Calling Stored Routines

When calling a stored routine, you must not use the same routine to calculate argument values by a stored function. For example, if the routine being called is also called by a stored function during the calculation of an argument value, passed arguments to the routine may be incorrect.

The following example shows a stored procedure P being called during the calculation of the arguments for another invocation of the stored procedure P:

```
SQL> CREATE MODULE M
cont>     LANG SQL
cont>
cont>     PROCEDURE P (IN :A INTEGER, IN :B INTEGER, OUT :C INTEGER);
cont>     BEGIN
cont>     SET :C = :A + :B;
cont>     END;
cont>
cont>     FUNCTION F () RETURNS INTEGER
cont>     COMMENT IS 'expect F to always return 2';
cont>     BEGIN
cont>     DECLARE :B INTEGER;
cont>     CALL P (1, 1, :B);
cont>     TRACE 'RETURNING ', :B;
cont>     RETURN :B;
cont>     END;
cont> END MODULE;
SQL>
SQL> SET FLAGS 'TRACE';
SQL> BEGIN
cont> DECLARE :CC INTEGER;
cont> CALL P (2, F(), :CC);
cont> TRACE 'Expected 4, got ', :CC;
cont> END;
~Xt: returning 2
~Xt: Expected 4, got 3
```

The result as shown above is incorrect. The routine argument values are written to the called routine's parameter area before complex expression values are calculated. These calculations may (as in the example) overwrite previously copied data.

The workaround is to assign the argument expression (in this example calling the stored function F) to a temporary variable and pass this variable as the input for the routine. The following example shows the workaround:

```
SQL> BEGIN
cont> DECLARE :BB, :CC INTEGER;
cont> SET :BB = F();
cont> CALL P (2, :BB, :CC);
cont> TRACE 'Expected 4, got ', :CC;
cont> END;
~Xt: returning 2
~Xt: Expected 4, got 4
```

This problem will be corrected in a future version of Oracle Rdb7.

5.0.57 Nested Correlated Subquery Outer References Incorrect

This problem was corrected in Oracle Rdb7 Release 7.0.0.2. An updated release note stating that this was fixed was inadvertently left out of all the following sets of release notes. Please note that this issue is now corrected. Outer references from aggregation subqueries contained within nested queries could receive incorrect values, causing the overall query to return incorrect results. The general symptom for an outer query that returned rows 1 to n was that the inner aggregation query would operate with the n^{th} - 1 row data (usually NULL for row 1) when it should have been using the n^{th} row data.

This problem has existed in various forms for all previous versions of Oracle Rdb7, but only appears in Release 6.1 and later when the inner of the nested queries contains an UPDATE statement.

The following example demonstrates the problem:

```
SQL> ATTACH 'FILENAME SHIPPING';
SQL> SELECT * FROM MANIFEST WHERE VOYAGE_NUM = 4904 OR
cont>                             VOYAGE_NUM = 4909;
  VOYAGE_NUM       EXP_NUM   MATERIAL              TONNAGE
        4904           311   CEDAR                  1200
        4904           311   FIR                     690
        4909           291   IRON ORE               3000
        4909           350   BAUXITE                1100
        4909           350   COPPER                 1200
        4909           355   MANGANESE               550
        4909           355   TIN                     500
7 rows selected

SQL> BEGIN
cont> FOR :A AS EACH ROW OF
cont>  SELECT * FROM VOYAGE V WHERE V.SHIP_NAME = 'SANDRA C.' OR
cont>                               V.SHIP_NAME = 'DAFFODIL' DO
cont>   FOR :B AS EACH ROW OF TABLE CURSOR MODCUR1 FOR
cont>    SELECT * FROM  MANIFEST M WHERE M.VOYAGE_NUM = :A.VOYAGE_NUM DO
cont>     UPDATE MANIFEST
cont>      SET TONNAGE = (SELECT (AVG (M1.EXP_NUM) *3) FROM MANIFEST M1
cont>                      WHERE M1.VOYAGE_NUM = :A.VOYAGE_NUM)
cont>      WHERE CURRENT OF MODCUR1;
cont>   END FOR;
cont> END FOR;
cont> END;
SQL> SELECT * FROM MANIFEST WHERE VOYAGE_NUM = 4904 OR
cont>                             VOYAGE_NUM = 4909;
  VOYAGE_NUM       EXP_NUM   MATERIAL              TONNAGE
        4904           311   CEDAR                     NULL
        4904           311   FIR                       NULL
        4909           291   IRON ORE                   933
        4909           350   BAUXITE                    933
        4909           350   COPPER                     933
        4909           355   MANGANESE                  933
        4909           355   TIN                        933
7 rows selected
```

The correct value for TONNAGE on both rows for VOYAGE_NUM 4904 (outer query row 1) is AVG (311+311)*3=933. However, Oracle Rdb7 calculates it as AVG (NULL+NULL)*3=NULL. In addition, the TONNAGE value for VOYAGE_NUM 4909 (outer query row 2) is actually the TONNAGE value for outer query row 1.

A workaround is to declare a variable of the same type as the outer reference data item, assign the outer reference data into the variable before the inner query that contains the correlated aggregation subquery, and reference the variable in the aggregation subquery. Keep in mind the restriction on the use of local variables in FOR cursor loops.

For example:

```
SQL> DECLARE :VN INTEGER;
SQL> BEGIN
cont> FOR :A AS EACH ROW OF
cont>  SELECT * FROM VOYAGE V WHERE V.SHIP_NAME = 'SANDRA C.' DO
cont>   SET :VN = :A.VOYAGE_NUM;
cont>   FOR :B AS EACH ROW OF TABLE CURSOR MODCUR1 FOR
cont>    SELECT * FROM MANIFEST M WHERE M.VOYAGE_NUM = :A.VOYAGE_NUM DO
cont>     UPDATE MANIFEST
cont>      SET TONNAGE = (SELECT (AVG (M1.EXP_NUM) *3) FROM MANIFEST M1
cont>                      WHERE M1.VOYAGE_NUM = :VN)
cont>      WHERE CURRENT OF MODCUR1;
cont>   END FOR;
cont> END FOR;
cont> END;
SQL> SELECT * FROM MANIFEST WHERE VOYAGE_NUM = 4904;
  VOYAGE_NUM       EXP_NUM   MATERIAL              TONNAGE
        4904           311   CEDAR                     933
        4904           311   FIR                       933
```

This problem was corrected in Oracle Rdb7 Release 7.0.0.2. An updated release note stating that this was fixed was inadvertently left out of all the following sets of release notes. Please note that this issue is now corrected.

5.0.58 Considerations when Using Holdable Cursors

If your applications use holdable cursors, be aware that after a COMMIT or ROLLBACK statement is executed, the result set selected by the cursor may not remain stable. That is, rows may be inserted, updated, and deleted by other users because no locks are held on the rows selected by the holdable cursor after a commit or rollback occurs. Moreover, depending on the access strategy, rows not yet fetched may change before Oracle Rdb actually fetches them.

As a result, you may see the following anomalies when using holdable cursors in a concurrent user environment:

- If the access strategy forces Oracle Rdb to take a data snapshot, the data read and cached may be inaccurate by the time the cursor fetches the data.

  For example, user 1 opens a cursor and commits the transaction. User 2 deletes rows read by user 1 (this is possible because the read locks are released). It is possible for user 1 to report data now deleted and committed.

- If the access strategy uses indexes that allow duplicates, updates to the duplicates chain may cause rows to be skipped, or even revisited.

  Oracle Rdb keeps track of the dbkey in the duplicate chain pointing to the data that was fetched. However, the duplicates chain could be revised by the time Oracle Rdb returns to using it.

Holdable cursors are a very powerful feature for read-only or predominantly read-only environments. However, in concurrent update environments, the instability of the cursor may not be acceptable. The stability of holdable cursors for update environments will be addressed in future versions of Oracle Rdb.

You can define the logical name RDMS$BIND_HOLD_CURSOR_SNAP or configuration parameter RDB_BIND_HOLD_CURSOR_SNAP to the value 1 to force all hold cursors to fetch the result set into a cached data area. (The cached data area appears as a "Temporary Relation" in the optimizer strategy displayed by the SET FLAGS STRATEGY statement or the RDMS$DEBUG_FLAGS S flag.) This logical name or configuration parameter helps to stabilize the cursor to some degree.

5.0.59 INCLUDE SQLDA2 Statement Is Not Supported for SQL Precompiler for PL/I in Oracle Rdb Release 5.0 or Higher

The SQL statement INCLUDE SQLDA2 is not supported for use with the PL/I precompiler in Oracle Rdb Release 5.0 or higher.

There is no workaround. This problem will be fixed in a future version of Oracle Rdb.

5.0.60 SQL Pascal Precompiler Processes ARRAY OF RECORD Declarations Incorrectly

The Pascal precompiler for SQL gives an incorrect %SQL-I-UNMATEND error when it parses a declaration of an array of records. The precompiler does not associate the END statement with the record definition, and the resulting confusion in host variable scoping causes a fatal error.

A workaround for the problem is to declare the record as a type and then define your array of that type. For example:

```
    main.spa:

        program main (input,output);

        type
        exec sql include 'bad_def.pin';    !gives error
        exec sql include 'good_def.pin';   !ok
        var
            a : char;

        begin
        end.

-----------------------------------------------------------------
    bad_def.pin

    x_record = record
    y  : char;
    variable_a:  array [1..50] of record
                a_fld1 : char;
                b_fld2  : record;
                            t : record
                                    v : integer;
                            end;
                end;
        end;
    end;
 -----------------------------------------------------------------

    good_def.pin
```

```
good_rec = record
        a_fld1 : char;
        b_fld2 : record
                 t : record
                         v: integer;
                 end;
        end;
end;

    x_record = record
        y  : char
        variable_a : array [1..50] of good_rec;
    end;
```

5.0.61 RMU Parallel Backup Command Not Supported for Use with SLS

The RMU Parallel Backup command is not supported for use with the Storage Library System (SLS) for OpenVMS.

5.1 Oracle CDD/Repository Restrictions

This section describes known problems and restrictions in Oracle CDD/Repository Release 7.0 and earlier.

5.1.1 Oracle CDD/Repository Compatibility with Oracle Rdb Features

Some Oracle Rdb features are not fully supported by all versions of Oracle CDD/Repository. Table 5–1 shows which versions of Oracle CDD/Repository support Oracle Rdb features and the extent of support.

In Table 5–1, repository support for Oracle Rdb7 features can vary as follows:

- Explicit support—The repository recognizes and integrates the feature, and you can use the repository to manipulate the item.
- Implicit support—The repository recognizes and integrates the feature, but you cannot use any repository interface to manipulate the item.
- Pass-through support—The repository does not recognize or integrate the feature, but allows the Oracle Rdb7 operation to complete without aborting or overwriting metadata. With pass-through support, a CDD-I-MBLRSYNINFO informational message may be returned.

Table 5–1 Oracle CDD/Repository Compatibility for Oracle Rdb Features

Oracle Rdb Feature	Minimum Release of Oracle Rdb	Minimum Release of Oracle CDD/Repository	Support
CASE, NULLIF, and COALESCE expressions	6.0	6.1	Implicit
CAST function	4.1	7.0	Explicit
Character data types to support character sets	4.2	6.1	Implicit
Collating sequences	3.1	6.1	Explicit
Constraints (PRIMARY KEY, UNIQUE, NOT NULL, CHECK, FOREIGN KEY)	3.1	5.2	Explicit

(continued on next page)

Table 5–1 (Cont.) Oracle CDD/Repository Compatibility for Oracle Rdb Features

Oracle Rdb Feature	Minimum Release of Oracle Rdb	Minimum Release of Oracle CDD/Repository	Support
CURRENT_DATE, CURRENT_ TIME, and CURRENT_ TIMESTAMP functions	4.1	7.0	Explicit
CURRENT_USER, SESSION_ USER, SYSTEM_USER functions	6.0	7.0	Explict
Date arithmetic	4.1	6.1	Pass-through
DATE ANSI, TIME, TIMESTAMP, and INTERVAL data types	4.1	6.1	Explicit
Delimited identifiers	4.2	6.1[1]	Explicit
External functions	6.0	6.1	Pass-through
External procedures	7.0	6.1	Pass-through
EXTRACT, CHAR_LENGTH, and OCTET_LENGTH functions	4.1	6.1	Explicit
GRANT/REVOKE privileges	4.0	5.0 accepts but does not store information	Pass-through
Indexes	1.0	5.2	Explicit
INTEGRATE DOMAIN	6.1	6.1	Explicit
INTEGRATE TABLE	6.1	6.1	Explicit
Logical area thresholds for storage maps and indexes	4.1	5.2	Pass-through
Multinational character set	3.1	4.0	Explicit
Multiversion environment (multiple Rdb versions)	4.1	5.1	Explicit
NULL keyword	2.2	7.0	Explicit
Oracle7 compatibility functions, such as CONCAT, CONVERT, DECODE, and SYSDATE	7.0	7.0	Explicit
Outer joins, derived tables	6.0	7.0	Pass-through
Query outlines	6.0	6.1	Pass-through
Storage map definitions correctly restored	3.0	5.1	Explicit
Stored functions	7.0	6.1	Pass-through
Stored procedures	6.0	6.1	Pass-through
SUBSTRING function	4.0	7.0 supports all features 5.0 supports all but 4.2 MIA features [2]	Explicit
Temporary tables	7.0	6.1	Pass-through
Triggers	3.1	5.2	Pass-through

[1]The repository does not preserve the distinction between uppercase and lowercase identifiers. If you use delimited identifiers with Oracle Rdb, the repository ensures that the record definition does not include objects with names that are duplicates except for case.

[2]Multivendor Integration Architecture (MIA) features include the CHAR_LENGTH clause and the TRANSLATE function.

(continued on next page)

Table 5–1 (Cont.) Oracle CDD/Repository Compatibility for Oracle Rdb Features

Oracle Rdb Feature	Minimum Release of Oracle Rdb	Minimum Release of Oracle CDD/Repository	Support
TRUNCATE TABLE	7.0	6.1	Pass-through
TRIM and POSITION functions	6.1	7.0	Explicit
UPPER, LOWER, TRANSLATE functions	4.2	7.0	Explicit
USER function	2.2	7.0	Explict

5.1.2 Multischema Databases and CDD/Repository

You cannot use multischema databases with CDD/Repository and Oracle Rdb release 7.0 and earlier. This problem will be corrected in a future release of Oracle Rdb.

5.1.3 Interaction of Oracle CDD/Repository Release 5.1 and Oracle RMU Privileges Access Control Lists

Oracle Rdb provides special Oracle RMU privileges that use the unused portion of the OpenVMS access control list (ACL) to manage access to Oracle RMU operations.

You can use the RMU Set Privilege and RMU Show Privilege commands to set and show the Oracle RMU privileges. The DCL SHOW ACL and DIRECTORY/ACL commands also show the added access control information; however, these tools cannot translate the names defined by Oracle Rdb.

Note

The RMU Convert command propagates the database internal ACL to the root file for access control entries (ACEs) that possess the SECURITY and DBADM (ADMINISTRATOR) privileges.

Oracle CDD/Repository protects its repository (dictionary) by placing the CDD$SYSTEM rights identifier on each file created within the anchor directory. CDD$SYSTEM is a special, reserved rights identifier created by Oracle CDD/Repository.

When Oracle CDD/Repository executes the DEFINE REPOSITORY command, it adds (or augments) an OpenVMS default ACL to the anchor directory. Typically, this ACL allows access to the repository files for CDD$SYSTEM and denies access to everyone else. All files created in the anchor directory inherit this default ACL, including the repository database.

Unfortunately, there is an interaction between the default ACL placed on the repository database by Oracle CDD/Repository and the Oracle RMU privileges ACL processing.

Within the ACL on the repository database, the default access control entries (ACEs) that were inherited from the anchor directory will precede the ACEs added by RMU Restore. As a result, the CDD$SYSTEM identifier will not have any Oracle RMU privileges granted to it. Without these privileges, if the user does not have the OpenVMS SYSPRV privilege enabled, Oracle RMU operations, such as Convert and Restore, will not be allowed on the repository database.

The following problems may be observed by users who do not have the SYSPRV privilege enabled:

- While executing a CDO DEFINE REPOSITORY or DEFINE DICTIONARY command:
  - If the CDD$TEMPLATEDB backup (.rbf) file was created by a previous version of Oracle Rdb7, the automatic RMU Convert operation that will be carried out on the .rbf file will fail because SYSPRV privilege is required.
  - If the CDD$TEMPLATEDB backup (.rbf) file was created by the current version of Oracle Rdb7, the restore of the repository database will fail because the default ACEs that already existed on the repository file that was backed up will take precedence, preventing RMU$CONVERT and RMU$RESTORE privileges from being granted to CDD$SYSTEM or the user.
  - If no CDD$TEMPLATEDB is available, the repository database will be created without a template, inheriting the default ACL from the parent directory. The ACE containing all the required Oracle RMU privileges will be added to the end of the ACL; however, the preexisting default ACEs will prevent any Oracle RMU privilege from being granted.
- You must use the RMU Convert command to upgrade the database disk format to Oracle Rdb7 after installing Release 7.0. This operation requires the SYSPRV privilege.

  During the conversion, RMU Convert adds the ACE containing the Oracle RMU privileges at the end of the ACL. Because the repository database already has the default Oracle CDD/Repository ACL associated with it, the Oracle CDD/Repository ACL will take precedence, preventing the granting of the Oracle RMU privileges.
- During a CDO MOVE REPOSITORY command, the Oracle RMU privilege checking may prevent the move, as the RMU$COPY privilege has not been granted on the repository database.
- When you execute the CDD template builder CDD_BUILD_TEMPLATE, the step involving RMU Backup privilege has not been granted.

Oracle CDD/Repository Releases 5.2 and higher correct this problem. A version of the Oracle CDD/Repository software that corrects this problem and allows new repositories to be created using Oracle Rdb7 is provided on the Oracle Rdb7 kit for use on OpenVMS VAX systems. See Section 5.1.3.1 for details.

5.1.3.1 Installing the Corrected CDDSHR Images

OpenVMS VAX Systems

Note

The following procedure must be carried out if you have installed or plan to install Oracle Rdb7 and have already installed CDD/Repository Release 5.1 software on your system.

Due to the enhanced security checking associated with Oracle RMU commands in Oracle Rdb on OpenVMS VAX, existing CDDSHR images for CDD/Repository Release 5.1 must be upgraded to ensure that the correct Oracle RMU privileges are applied to newly created or copied repository databases.

Included in the Oracle Rdb7 for OpenVMS VAX distribution kit is a CDD upgraded image kit, called CDDRDB042, that must be installed after you have installed the Oracle Rdb7 for OpenVMS VAX kit.

This upgrade kit should be installed by using VMSINSTAL. It automatically checks which version of CDDSHR you have installed and replaces the existing CDDSHR.EXE with the corrected image file. The existing CDDSHR.EXE will be renamed SYS$LIBRARY:OLD_CDDSHR.EXE.

The upgrade installation will also place a new CDD_BUILD_TEMPLATE.COM procedure in SYS$LIBRARY for use with CDD/Repository V5.1.

Note

If you upgrade your repository to CDD/Repository V5.1 after you install Oracle Rdb7 V7.0, you must install the corrected CDDSHR image again to ensure that the correct CDDSHR images have been made available.

The CDD/Repository upgrade kit determines which version of CDD/Repository is installed and replaces the existing CDDSHR.EXE with the appropriate version of the corrected image.

5.1.3.2 CDD Conversion Procedure

OpenVMS VAX Systems

Oracle Rdb7 provides RDB$CONVERT_CDD$DATABASE.COM, a command procedure that both corrects the anchor directory ACL and performs the RMU Convert operation. The command procedure is located in SYS$LIBRARY.

Note

You must have SYSPRV enabled before you execute the procedure RDB$CONVERT_CDD$DATABASE.COM because the procedure performs an RMU Convert operation.

Use the procedure RDB$CONVERT_CDD$DATABASE.COM to process the anchor directory and update the ACLs for both the directory and, if available, the repository database.

This procedure accepts one parameter: the name of the anchor directory that contains, or will contain, the repository files. For example:

```
$ @SYS$LIBRARY:DECRDB$CONVERT_CDD$DATABASE [PROJECT.CDD_REP]
```

If many repositories exist on a system, you may want to create a DCL command procedure to locate them, set the Oracle RMU privileges ACL, and convert the databases. Use DCL commands similar to the following:

```
$ LOOP:
$       REP_SPEC = F$SEARCH("[000000...]CDD$DATABASE.RDB")
$       IF REP_SPEC .NES. ""
$       THEN
$           @SYS$LIBRARY:DECRDB$CONVERT_CDD$DATABASE    -
                'F$PARSE(REP_SPEC,,,"DIRECTORY")'
$           GOTO LOOP
$       ENDIF
```