Oracle® Rdb for OpenVMS

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Oracle® Rdb for OpenVMS

Release Notes

Release 7.2.1.0

January 2007

Oracle Rdb Release Notes, Release 7.2.1.0 for OpenVMS

Contents

Preface

Purpose of This Manual

This manual contains release notes for Oracle Rdb Release 7.2.1.0. The notes describe changed and enhanced features; upgrade and compatibility information; new and existing software problems and restrictions; and software and documentation corrections.

Intended Audience

This manual is intended for use by all Oracle Rdb users. Read this manual before you install, upgrade, or use Oracle Rdb Release 7.2.1.0.

Document Structure

This manual consists of the following chapters:

Chapter 1	Describes how to install Oracle Rdb Release 7.2.1.0.
Chapter 2	Describes problems corrected in Oracle Rdb Release 7.2.1.0.
Chapter 3	Describes problems corrected in Oracle Rdb Release 7.2.0.2.
Chapter 4	Describes problems corrected in Oracle Rdb Release 7.2.0.1.
Chapter 5	Describes problems corrected in Oracle Rdb Release 7.2.
Chapter 6	Describes enhancements introduced in Oracle Rdb Release 7.2.1.0.
Chapter 7	Describes enhancements introduced in Oracle Rdb Release 7.2.0.2.
Chapter 8	Describes enhancements introduced in Oracle Rdb Release 7.2.0.1.
Chapter 9	Describes enhancements introduced in Oracle Rdb Release 7.2.
Chapter 10	Describes problems, restrictions, and workarounds known to exist in Oracle Rdb Release 7.2.1.0.

Chapter 1
Installing Oracle Rdb Release 7.2.1.0

This software update is installed using the OpenVMS VMSINSTAL utility.

NOTE

Oracle Rdb Release 7.2 kits are full kits. There is no requirement to install any prior release of Oracle Rdb when installing new Rdb Release 7.2 kits.

1.1 Oracle Rdb on HP OpenVMS Industry Standard 64

The Oracle Rdb product family is available on the HP OpenVMS Industry Standard 64 platform and the OpenVMS AlphaServer platform. In general, the functionality present in Oracle Rdb on OpenVMS Alpha will be available in Oracle Rdb on OpenVMS Industry Standard 64. However, certain differences between the platforms may result in minor capability and functionality differences.

The database format for Oracle Rdb Release 7.2 is the same on both I64 and Alpha platforms and databases may be accessed simultaneously from both architectures in a cluster environment. Access to an Oracle Rdb Release 7.2 database from prior Rdb versions (on Alpha or VAX platforms) or from other systems on the network is available via the Oracle Rdb remote database server.

1.2 Requirements

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

- This Oracle Rdb release requires the following OpenVMS environments:
  - ♦ OpenVMS Alpha V8.2 to V8.3−x.
  - ♦ OpenVMS Industry Standard 64 V8.2−1 to V8.3−x.
- Oracle Rdb must be shutdown before you install this update kit. That is, the command file SYS$STARTUP:RMONSTOP72.COM should be executed before proceeding with this installation. If you have an OpenVMS cluster, you must shutdown the Rdb Release 7.2 monitor on all nodes in the cluster before proceeding.
- The installation requires approximately 280,000 blocks for OpenVMS Alpha systems.
- The installation requires approximately 500,000 blocks for OpenVMS I64 systems.
- Oracle strongly recommends that all available OpenVMS patches are installed on all systems prior to installing Oracle Rdb Release 7.2.1. Contact your HP support representitive for more information and assistance.

1.3 Maximum OpenVMS Version Check

OpenVMS Version 8.3−x is the maximum supported version of OpenVMS for this release of Oracle Rdb.

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.

1.4 Database Format Changed

The Oracle Rdb on−disk database format has been incremented to 721. An RMU /CONVERT operation is required for databases created by or accessed by Oracle Rdb V7.0 or V7.1 to be accessed with Rdb Release 7.2.

Prior to upgrading to Oracle Rdb Release 7.2 and prior to converting an existing database to Oracle Rdb Release 7.2 format, Oracle strongly recommends that you perform a full database verification (with the "RMU /VERIFY /ALL" command) along with a full database backup (with the "RMU /BACKUP" command) to ensure a valid and protected database copy.

1.5 Using Databases from Releases Earlier than V7.0

You cannot convert or restore databases earlier than the Oracle Rdb V7.0 format directly to Oracle Rdb V7.2 format. The RMU Convert command for Oracle Rdb V7.2 supports conversions from Oracle Rdb V7.0 and V7.1 format databases only. If you have an Oracle Rdb V3.0 through V6.1 format database or database backup, you must convert it to at least Oracle Rdb V7.0 format and then convert it to Oracle Rdb V7.2 format. For example, if you have a V4.2 format database, you must convert it first to at least Oracle Rdb V7.0 format, then convert it to Oracle Rdb V7.2 format.

If you attempt to convert or restore a database that is prior to Oracle Rdb V7.0 format directly to Oracle Rdb V7.2 format, Oracle RMU generates an error.

1.6 Invoking the VMSINSTAL Procedure

The installation procedure for Oracle Rdb has been simplified. All Oracle Rdb components are always installed and the number of prompts during the installation has been reduced. The installation procedure is the same for Oracle Rdb for OpenVMS Alpha and Oracle Rdb for OpenVMS I64.

To start the installation procedure, invoke the VMSINSTAL command procedure as in the following examples.

- To install the Oracle Rdb for OpenVMS I64 kit that is performance targeted for I64 platforms:

  ```
  @SYS$UPDATE:VMSINSTAL RDBV72100IM device−name
  ```
- To install the Oracle Rdb for OpenVMS Alpha kit that is compiled to run on all Alpha platforms:

  ```
  @SYS$UPDATE:VMSINSTAL RDBV72100AM device−name
  ```
- To install the Oracle Rdb for OpenVMS Alpha kit that is performance targeted for Alpha EV56 and later platforms:

  ```
  @SYS$UPDATE:VMSINSTAL RDBV72101AM device−name
  ```

device−name

Use the name of the device on which the media is mounted. If the device is a disk−type drive, you also need to specify a directory. For example: *DKA400:[RDB.KIT]*

1.7 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 Rdb installed will probably not be usable.

1.8 After Installing Oracle Rdb

This update provides a new Oracle TRACE facility definition for Oracle Rdb. Any Oracle TRACE selections that reference Oracle Rdb will need to be redefined to reflect the new facility version number for the updated Oracle Rdb facility definition, "RDBVMSV7.2".

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

The installation procedure inserts the Oracle Rdb facility definition into a library file called EPC$FACILITY.TLB. To be able to collect Oracle Rdb 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=RDBVMSV7.2 −
_$ /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 the process executing the INSERT DEFINITION command must use the version of Oracle Rdb that matches the version used to create the Oracle TRACE administration database or the INSERT DEFINITION command will fail.

1.9 VMS$MEM_RESIDENT_USER Rights Identifier Required

Oracle Rdb Version 7.1 introduced additional privilege enforcement for the database or row cache attributes RESIDENT, SHARED MEMORY IS SYSTEM and LARGE MEMORY IS ENABLED. If a database utilizes any of these features, then the user account that opens the database must be granted the VMS$MEM_RESIDENT_USER rights identifier.

Oracle recommends that the RMU/OPEN command be used when utilizing these features.

1.10 Installation, Configuration, Migration, Upgrade Suggestions

Oracle Rdb Release 7.2 fully supports mixed−architecture clusters for AlphaServer systems and HP Integrity servers.

In certain development environments, it may be helpful to incorporate a VAX system into the AlphaServer systems and HP Integrity servers cluster. While HP and Oracle believe that in most cases this will not cause problems to the computing environment, we have not tested it extensively enough to provide support. It is possible that VAX systems in a cluster may cause a problem with the cluster performance or stability. Should this happen, the VAX systems in the cluster which are causing the difficulty should be removed.

Oracle continues to support mixed architecture clusters of VAX systems and AlphaServer systems with direct database access using Rdb V7.0. Oracle Rdb V7.1 runs natively on Alpha systems and clusters. All Rdb versions include a built−in remote network database server allowing cross−architecture and cross−version application and database access.

When moving applications from existing Alpha or VAX configurations to new enviroments containing Integrity Server systems, there are numerous possible paths depending on the requirements of individual sites. In general, this can be as straightforward as adding a new node to an already existing AlphaServer systems cluster or standalone system, except the node is an HP Integrity server. Table 1−1, Migration Suggestions, considers several possible situations and recommended steps to take.

Table 1−1 Migration Suggestions

Case	You Wish To...	You should...
1	Add an Integrity server to an existing cluster of Alpha servers	1. Verify database(s) using RMU/VERIFY/ALL. 2. Backup database(s) using RMU/BACKUP. 3. Install Rdb 7.2 on Integrity and Alpha nodes. 4. Convert database(s) to the Rdb 7.2 structure level using RMU/CONVERT. 5. Verify database(s) again using RMU/VERIFY/ALL. 6. Backup database(s) using RMU/BACKUP. 7. Access database(s) from Alpha and Integrity directly by specifying database root file specification(s) in SQL ATTACH statements.
2	Add an Integrity server to an existing mixed cluster of VAX and Alpha nodes and access an Rdb database	1. Verify database(s) using

	from all nodes. Disks used for the database are accessible from all nodes.	RMU/VERIFY/ALL. 2. Backup database(s) using RMU/BACKUP. 3. Install Rdb 7.2 on Integrity and Alpha nodes. 4. Convert database(s) to the Rdb 7.2 structure level using RMU/CONVERT. 5. Verify database(s) again using RMU/VERIFY/ALL. 6. Backup database(s) using RMU/BACKUP. 7. Access database(s) from Alpha and Integrity nodes directly by specifying database root file specification(s) in SQL ATTACH statements. 8. Access the database from VAX node(s) using the Rdb built−in network server (remote database) by specifying one of the Alpha or Integrity node names in SQL ATTACH statements. 9. After thorough testing, remove VAX nodes from the cluster.
3	Move database(s) to new disks and add an Integrity server to an existing cluster.	1. Use RMU/COPY with an options file to move the database files to the new disks. 2. Follow the steps for case 1 or case 2.
4	Continue to use Rdb primarily from VAX or Alpha nodes using earlier releases. Add an Integrity server for application testing purposes.	1. Install Rdb 7.2 on Integrity node. 2. Access existing database(s) from Integrity node by specifying one of the Alpha or VAX node names in the SQL ATTACH statements. 3. When testing is complete, follow the steps in case 1 or case 2.
5	Add an Integrity server to an existing cluster of Alpha servers or Create a new cluster from an existing stand−alone Alpha server by adding one or more new Integrity servers.	1. Verify database(s) using RMU/VERIFY/ALL. 2. Backup database(s) using RMU/BACKUP. 3. Install Rdb 7.2 on Integrity and Alpha nodes.

		4. Convert database(s) to the Rdb 7.2 structure level using RMU/CONVERT. 5. Verify database(s) again using RMU/VERIFY/ALL. 6. Backup database(s) using RMU/BACKUP. 7. Access database(s) from Alpha and Integrity directly by specifying database root file specification in the SQL ATTACH statements.
6	Create a new stand−alone Integrity Server system or cluster of Integrity Servers and move database(s) to the new environment.	1. Verify database(s) using RMU/VERIFY/ALL. 2. Install Rdb 7.2 on new system(s). 3. Back up database(s) on the existing cluster using RMU/BACKUP. 4. Copy backup file(s) to the new system (or, if using tape media, make the tapes available to the new system). 5. Restore database(s) on the new system using RMU/RESTORE specifying the location of each database file in an options file. 6. Verify the new database using RMU/VERIFY/ALL.

Refer to the Oracle Rdb documentation set for additional information and detailed instructions for using RMU and remote databases.

Note that database parameters might need to be altered in the case of accessing a database from a larger number of systems in a cluster.

Chapter 2
Software Errors Fixed in Oracle Rdb Release 7.2.1.0

This chapter describes software errors that are fixed by Oracle Rdb Release 7.2.1.0.

2.1 Software Errors Fixed That Apply to All Interfaces

2.1.1 Incorrect Backup Checksum and CRC Values on I64

In some cases, the checksum or CRC values within an .RBF backup file on I64 systems starting with Rdb Release V7.2.0.2 may be incorrect. This difference could result in checksum errors during restore operations when using /CRC=CHECKSUM. In other words, the restore on Rdb 7.2.0.2 I64 cannot read backups made by any other platform or version when CRC=CHECKSUM was used.

As a workaround, Oracle recommends using the default CRC algorithm of /CRC=AUTODIN_II rather than /CRC=CHECKSUM.

This problem has been corrected in Oracle Rdb Release 7.2.1. The checksum value calculated by Oracle Rdb is now the same on all platforms and versions.

2.1.2 Bugcheck Loop Created Many Bugcheck Dumps

Bugs 5411895 and 5361954

In some rare circumstances, a bugcheck dump could cause another bugcheck dump to occur, resulting in what would be an infinite number of bugcheck dumps, except that a finite number would, in fact, be produced. The limit to the number of bugcheck dumps being created was due to using up available disk space.

An attempt to remedy this issue has been made. After a process has created three bugcheck dumps, any further dumps should become "mini" dumps. After three mini bugcheck dumps, the dumps should cease and a COSI$_FATINTERR status returned.

All further bugcheck dump attempts should simply return COSI$_FATINTERR (in other words, no more bugcheck dumps will be produced).

An example of a query causing this problem is shown below.

```
SELECT   TIPO, OPFU
FROM     T1
WHERE    CODE  ='100' AND
         VALUE = '0057984193004785667' AND
         COOP  = 'OCA0604WOREST' AND
         CRTC = 'MFV' AND
         OPFU  = '20011228' AND
       ( TIPO  = 'TCI' OR
         (IVSN = 'I' AND (TIPO = 'COM' OR TIPO  = 'ATR') ) ) ;
%RDMS-I-BUGCHKDMP, generating bugcheck dump file RDSBUGCHK.DMP;
%RDMS-I-BUGCHKDMP, generating bugcheck dump file RDSBUGCHK.DMP;
%RDMS-I-BUGCHKDMP, generating bugcheck dump file RDSBUGCHK.DMP;
%RDMS-I-BUGCHKDMP, generating bugcheck dump file RDSBUGCHK.DMP;
%RDMS-I-BUGCHKDMP, generating bugcheck dump file RDSBUGCHK.DMP;
...etc...
```

This change has been made in Oracle Rdb Release 7.2.1.

2.1.3 Left Outer Join Query Slows With Full Index Scan

Bug 5567495

A customer reported that a particular query runs significantly slower after upgrading from Release 7.1.4.1 to Release 7.1.4.4 when using full index scan in the outer leg of the Left Outer join operation. We were able to reproduce the problem using the following simple script without data, since the difference in the output strategy is indicative of the problem.

```
create table t1 (
    col_one integer,
    col_two integer,
    price   integer);

create table t2 (
    col_one     integer,
    col_two     integer,
    line_amt    COMPUTED BY
    (select c1.price from t1 c1
     where ((c1.col_one = t2.col_one)
         and (c1.col_two = t2.col_two)))) ;

create unique index t1_ndx on t1 (col_one, col_two);
create unique index t2_ndx on t2 (col_one, col_two);

create view v_t1_loj_t2
    (col_one, col_two, total) as
    (select c2.col_one, c2.col_two, SUM(c3.line_amt)
    from t1 as c2
       LEFT OUTER JOIN
         t2 as c3
            ON ((c3.col_one = c2.col_one) AND
  (c3.col_two = c2.col_two))
    GROUP BY c2.col_one, c2.col_two);

! The following is the query that slows down in performance since the
! strategy applies a full index scan at the outer leg of the left outer join
! operation, as compared to "Direct lookup" index retrieval strategy.

select t1.col_two,
    (select v1.total from v_t1_loj_v2 v1
     where v1.col_one = t1.col_one and
            v1.col_two = t1.col_two) as v_total
from t1 where t1.col_one = 1 ;
Cross block of 2 entries
  Cross block entry 1
    Conjunct        Index only retrieval of relation T1
      Index name  T1_NDX [1:1]
  Cross block entry 2
    Aggregate       Conjunct        Aggregate
    Cross block of 2 entries
      Cross block entry 1
        Match    (Left Outer Join)
          Outer loop
            Index only retrieval of relation T1
              Index name  T1_NDX [0:0]               <== Full index scan
          Inner loop      (zig-zag)
            Index only retrieval of relation T2
              Index name  T2_NDX [0:0]
      Cross block entry 2
```

```
        Aggregate       Get     Retrieval by index of relation T1
          Index name  T1_NDX [2:2]   Direct lookup
0 rows selected
```

The strategy from Rdb Release 7.1.4.1.1 is exactly the same with the exception of the better performing index retrieval in the Outer loop of the Left Outer Join operation.

```
Cross block of 2 entries
  Cross block entry 1
    Conjunct        Index only retrieval of relation T1
      Index name  T1_NDX [1:1]
  Cross block entry 2
    Aggregate       Conjunct        Aggregate
    Cross block of 2 entries
      Cross block entry 1
        Match    (Left Outer Join)
          Outer loop
            Index only retrieval of relation T1
              Index name  T1_NDX [2:2]   Direct lookup      <== Best one
          Inner loop      (zig-zag)
            Index only retrieval of relation T2
              Index name  T2_NDX1 [0:0]
      Cross block entry 2
        Aggregate       Get     Retrieval by index of relation T1
          Index name  T1_NDX [2:2]   Direct lookup
```

There is no known workaround for this problem.

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

1. The main query selects from a table (T1) using the WHERE filter predicate on the column which is used as one of the join items in the subselect query to join another view.
2. The view is defined as an aggregate left outer join query between T1 and a second table (T2) on the two join columns.
3. The last column of the view is an aggregate function SUM on the COMPUTED BY column of the table T2 which contains a SELECT query to join table T1 using the same two columns.

This problem has been corrected in Oracle Rdb Release 7.2.1.

2.1.4 DBR Bugchecks at DIO$LACB_CREATE + 00000174

Bug 5467328

It was possible for the database recovery process (DBR) to bugcheck when a process terminated while rolling back the creation of a new table or index. For example, consider the following script:

```
SQL> CREATE DATABASE FILENAME TEST;
SQL> CREATE TABLE T1 (C1 INT);
SQL> INSERT INTO T1 VALUES (1);
1 row inserted
SQL> ROLLBACK;
```

In the above sequence of commands, if during the course of rolling back the transaction, the process were to rollback the creation of table T1, update the area−inventory page (AIP) for the table, and flush the changed AIP entry to disk, but get terminated before it could truncate the recovery−unit journal (RUJ), then the DBR

would fail when attempting to access the non−existant table T1. The first DBR bugcheck would contain an exception similar to the following:

```
***** Exception at 00000000001A6C0C : RDMDBR72\DIOLACB$LACB_AIP_ENT_GET +
000005CC
%COSI−F−BUGCHECK, internal consistency failure
```

Subsequent attempts to access the database would result in a DBR bugcheck with an exception similar to the following:

```
***** Exception at 00000000001A6174 : RDMDBR72\DIO$LACB_CREATE + 00000174
%RDMS−F−CANTFINDLAREA, cannot locate logical area 58 in area inventory page list
```

This problem would occur because the DBR was not prepared for the possibility that a table might not exist when it attempted to rollback inserts for the table.

If this problem is encountered, there is no supported workaround to resolve the problem. The database must be restored and recovered or Oracle Rdb must be updated to the release that contains the fix for this problem before attempting to access the database.

This problem has been corrected in Oracle Rdb Release 7.2.1.

2.1.5 Processes Do Not Always Terminate After Monitor Terminates

Bug 5361981

When the Oracle Rdb monitor process terminates abnormally, all user processes that are attached to databases on that node should immediately terminate. However, there were cases where that didn't happen, and those user processes would continue to access Oracle Rdb resources after the monitor failed. Consider the following example.

1. User 1, node 1: *SQL> ATTACH 'FILENAME MF_PERSONNEL';*
2. User 2, node 2: *SQL> ATTACH 'FILENAME MF_PERSONNEL';*
3. User 3, node 1: *$ STOP/ID={pid of monitor process on node 1}*

In the above sequence of events, the user process on node 1 should have terminated as soon as the monitor process was killed, but it remained active.

This problem can be avoided by using the RMU/OPEN command and manually opening databases on all nodes that will have users accessing the database.

This problem has been corrected in Oracle Rdb Release 7.2.1.

2.1.6 Wrong Results With "OR Index Retrieval" and Bitmapped Scan

Bug 5363170

When a strategy was chosen by the optimizer that included "OR Index retrieval" and bitmapped scan was enabled, queries could return incorrect results.

The following example should return 100 rows, but only returns one. Note that bitmap scan is enabled and the strategy includes "OR index retrieval" on the second (inner) cross block.

```
SQL> set flags 'bitmap,strategy,detail'
SQL> select t1.id from t1 join t2
cont>  on t1.id=t2.id or t1.id=t2.id
cont>  where t1.f2=1 and t1.f2=1;
Tables:
  0 = T1
  1 = T2
Cross block of 2 entries
  Cross block entry 1
    Conjunct: 0.F2 = 1
    Conjunct: 0.F2 = 1
    Get     Retrieval sequentially of relation 0:T1
  Cross block entry 2
    Conjunct: ((0.ID = 1.ID) OR (0.ID = 1.ID)) AND (0.F2 = 1) AND (0.F2 = 1)
               AND (0.ID = 1.ID)
    OR index retrieval
      Index only retrieval of relation 1:T2
        Index name  I2 [1:1]
          Keys: 0.ID = 1.ID
 T1.ID
 00164
1 row selected
```

The problem can be avoided by not using bitmap scan.

This problem has been corrected in Oracle Rdb Release 7.2.1.

2.1.7 Bugcheck in Query on a Single Table With AND/OR Predicates

Bug 5361954

The customer states that a specific query always bugchecks with the following query after migrating a database from Alpha Rdb 7.0.6.5 to Itanium 7.2.0.0.

```
SELECT   TIPO, OPFU
FROM     T1
WHERE    CODE  ='100' AND
         VALUE = '0057984193004785667' AND
         COOP  = 'OCA0604WOREST' AND
         CRTC = 'MFV' AND
         OPFU  = '20011228' AND
       ( TIPO  = 'TCI' OR (IVSN = 'I' AND (TIPO = 'COM' OR TIPO  = 'ATR') ) ) ;
%RDMS-I-BUGCHKDMP, generating bugcheck dump file RDSBUGCHK.DMP;
%RDB-F-BUG_CHECK, internal consistency check failed
```

The problem is not specific to the Rdb 7.2 Integrity platform. The bugchecks have been seen with all Rdb versions.

Sometimes the query no longer fails if the query is executed after a metadata query such as "select * from rdb$database". Performing an EXPORT/IMPORT also does not show the problem.

The query works if the SQL flags 'MAX_STABILITY' is defined, as in the following example.

```
SET FLAGS 'MAX_STABILITY';

SELECT   TIPO, OPFU
FROM     T1
WHERE    CODE  ='100' AND
         VALUE = '0057984193004785667' AND
         COOP  = 'OCA0604WOREST' AND
         CRTC  = 'MFV' AND
         OPFU  = '20011228' AND
       ( TIPO  = 'TCI' OR (IVSN = 'I' AND (TIPO = 'COM' OR TIPO  = 'ATR') ) ) ;
Firstn                    Sort                        Conjunct
Get                       Retrieval by index of relation T1
  Index name  T1_NDX3 [5:5,(7:7)2] Bool
 TIPO   OPFU
 COM    20011228
1 row selected
```

The following indices must be defined for the table T1 for the query to fail.

```
T1_NDX1 with column CODE
    and column NUMOP

T1_NDX2 with column TIPO
    and column CODE
    and column VALUE
    and column COOP
    and column CRTC
    and column OPFU
    and column IVSN
    and column NUMOP

T1_NDX3 with column CODE descending
    and column VALUE descending
    and column COOP
    and column CRTC
    and column TIPO
    and column IVSN
    and column OPFU
```

This problem has been corrected in Oracle Rdb Release 7.2.1.

2.1.8 Bugcheck When Bitmap Scan Enabled

Bug 5414635

The following query bugchecks when the bitmap scan feature is enabled.

```
set flags 'bitmap';
select * from T1 where C2 = 'XYZ' or C4 >='03-jul-2006' ;
%RDMS-I-BUGCHKDMP, generating bugcheck dump file RDSBUGCHK.DMP;
```

The query works if the bitmap scan is disabled, as in the following example.

```
set flags 'nobitmap';
SQL> select * from T1 where C2 = 'XYZ' or C4 >='03-jul-2006' ;
Tables:
  0 = T1
Conjunct: (0.C2 = 'XYZ') OR (0.C4 >= '3-JUL-2006')
Get     Retrieval by index of relation 0:T1
  Index name  X1_T1 [0:0]
0 rows selected
```

This problem has been corrected in Oracle Rdb Release 7.2.1.

2.1.9 Wrong Result From Star Join Query With Aggregate

Bug 5192800

The following query looks like a star join between the table t0 and two other tables (t1 and t2). It should return zero rows, but instead it incorrectly finds one row.

```
select t0.acct_num, t1.scard_id
  from
    subscr t0
    inner join
    scard t1
    on
        (t1.acct_num = t0.acct_num
         and exists
           (select *
             from hwatt t3 where t3.acct_num = t1.acct_num))
    left outer join
    service t2
    on
      t2.acct_num = t0.acct_num and
      t2.dt_tim_stmp =
            (select max(t4.dt_tim_stmp)
              from service t4
              where t4.acct_num = t0.acct_num);
Tables:
  0 = SUBSCR
  1 = SCARD
  2 = SERVICE
  3 = HWATT
  4 = SERVICE
Cross block of 2 entries
  Cross block entry 1
    Cross block of 2 entries        (Left Outer Join)
      Cross block entry 1
        Cross block of 3 entries
          Cross block entry 1
            Get     Retrieval sequentially of relation 0:SUBSCR
          Cross block entry 2
            Aggregate: 0:MAX (4.DT_TIM_STMP)
            Conjunct: 4.ACCT_NUM = 0.ACCT_NUM
            Get     Retrieval sequentially of relation 4:SERVICE
          Cross block entry 3
            Conjunct: 1.ACCT_NUM = 0.ACCT_NUM
            Get     Retrieval sequentially of relation 1:SCARD
      Cross block entry 2
        Conjunct: (2.ACCT_NUM = 0.ACCT_NUM) AND (2.DT_TIM_STMP = <agg0>)
        Get     Retrieval sequentially of relation 2:SERVICE
```

```
  Cross block entry 2
    Aggregate-F1: 1:COUNT-ANY (<subselect>)
    Conjunct: 3.ACCT_NUM = 1.ACCT_NUM
    Get     Retrieval sequentially of relation 3:HWATT
 T0.ACCT_NUM   T1.SCARD_ID
          1            2
1 row selected
```

The query works if the equality predicate with the MAX aggregate subselect query is converted into a WHERE clause, as in the following example.

```
select t0.acct_num, t1.scard_id
  from
    subscr t0
    inner join
    scard t1
    on
        (t1.acct_num = t0.acct_num
         and exists
           (select *
             from hwatt t3 where t3.acct_num = t1.acct_num))
    left outer join
    service t2
    on
      t2.acct_num = t0.acct_num
where ! <== converted into WHERE clause
      h.dt_tim_stmp =
            (select max(sv2.dt_tim_stmp)
              from service sv2 where sv2.acct_num = ss.acct_num);
Tables:
  0 = SUBSCR
  1 = SCARD
  2 = SERVICE
  3 = HWATT
  4 = SERVICE
Cross block of 2 entries
  Cross block entry 1
    Cross block of 2 entries        (Left Outer Join)
      Cross block entry 1
        Cross block of 3 entries
          Cross block entry 1
            Get     Retrieval sequentially of relation 0:SUBSCR
          Cross block entry 2
            Conjunct: 1.ACCT_NUM = 0.ACCT_NUM
            Get     Retrieval sequentially of relation 1:SCARD
          Cross block entry 3
            Conjunct: <agg0> <> 0                  <== See Note
            Aggregate-F1: 0:COUNT-ANY (<subselect>)
            Conjunct: 3.ACCT_NUM = 1.ACCT_NUM
            Get     Retrieval sequentially of relation 3:HWATT
      Cross block entry 2
        Conjunct: 2.ACCT_NUM = 0.ACCT_NUM
        Get     Retrieval sequentially of relation 2:SERVICE
  Cross block entry 2
    Conjunct: 2.DT_TIM_STMP = agg1
    Aggregate: 1:MAX (4.DT_TIM_STMP)
    Conjunct: 4.ACCT_NUM = 0.ACCT_NUM
    Get     Retrieval sequentially of relation 4:SERVICE
0 rows selected
```

Note:: The conjunct "<agg0> <> 0" appears in the good strategy output at the top of the Aggregate for EXISTS statement, but this conjunct is missing in the strategy output of the problem query.

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

1. The main query looks like a star join, joining a main table (t0) to multiple tables (in this case, two tables t1 and t2).
2. Table t0 and t1 are inner joined followed by an EXISTS subselect query referencing t3.
3. Table t0 and t2 are left outer joined followed by an equality predicate with a MAX aggregate subselect query which joins table t0 and t4.

This problem has been corrected in Oracle Rdb Release 7.2.1.

2.1.10 Restriction Relaxed for Executing External Routines via Privileged Images

Bugs 4595983 and 4606819

In prior versions of Oracle Rdb, an external routine could not be activated if it was called from an image that had more privileges than the current process was granted by OpenVMS. This security precaution prevented an unsafe sharable image being substituted in the production environment.

The following example shows the error reported by Oracle Rdb.

```
%RDB-E-EXTFUN_FAIL, external routine failed to compile or execute successfully
-RDMS-E-RTNUSENOTALL, routine "GET_ONE" can not be used, too many privileges
available
```

This restriction required SQL/Services applications to define external routines with BIND ON SERVER to allow the routine to be activated outside the privileged environment. This also causes problems for RMU Load (when constraints, triggers and AUTOMATIC INSERT columns call external routines), RMU Unload (when COMPUTED BY columns and views call external routines), and RMU Verify (which evaluated constraints).

Starting with Rdb V7.2, you can use the INSTALL ADD/SHARE command to make the external routine's sharable image "known". The sharable image must be activated by system wide logical names with the /EXECUTIVE mode option. Such images are assumed to be trusted in the case where a privileged image (such as RMU) executes the external routine.

In the following example, assume there is a function named GET_ONE in a shared image called FUNCS.EXE which is built from a single module named FUNCS.OBJ. The database has the following external function definition:

```
SQL> create function get_one() returns integer;
cont> external
cont>  name RETURN_ONE
cont>  location 'MY_DIR:FUNCS.EXE'
cont>  language general
cont>  parameter style general
cont>  deterministic
cont>  comment is 'Always returns 1';
```

Suppose there is a SQL$PRE/CC application named "PRIV_APP" which must be installed with extra privileges and which contains the following code:

```
EXEC SQL select get_one() into :result from rdb$database;
printf("%d is the loneliest number\n");
```

If FUNCS.EXE is not installed and PRIV_APP is executed from an account which has lower privileges than those given PRIV_APP when it was installed, the call to GET_ONE() shown above will fail with a RTNUSENOTALL error. In order to be able to execute GET_ONE() from within PRIV_APP, locate the sharable image in SYS$SHARE, and install FUNCS.EXE using DCL similar to the following.

```
$ LINK/SHARE/NOTRACE/EXE=SYS$COMMON:[SYSLIB]FUNCS.EXE FUNCS,SYS$INPUT:/OPTIONS
SYMBOL_VECTOR=(RETURN_ONE=PROCEDURE)
$ SET FILE /PROTECTION=W:RE SYS$SHARE:FUNCS.EXE
$ INSTALL ADD SYS$SHARE:FUNCS.EXE /SHARE
$ DEFINE/SYSTEM/EXECUTIVE_MODE MY_DIR SYS$SHARE
```

Using the installed sharable image PRIV_APP will now successfully execute the SQL statement that results in a successful call to the function GET_ONE.

This problem was corrected in Oracle Rdb Release 7.2.

2.1.11 RDMS$_ Symbols Missing From RDMMSGSHR on I64

Bug 5309253

In prior Oracle Rdb 7.2 releases on I64, all message symbols were erroneously omitted from RDMMSGSHR.EXE.

This problem has been corrected in Oracle Rdb Release 7.2.1.

2.1.12 Hangs or Looping When Lots of Page Contention

Applications that had lots of page contention could sometimes hang due to page locks not being released by a process or they could enter a CPU loop. This problem was only in Oracle Rdb Release 7.2.

This problem would occur when an internal queue used to manage blocking AST requests would become corrupt. In that situation, blocking ASTs could be lost or processing of the queue could result in an infinite loop.

There is no workaround for this problem.

This problem has been corrected in Oracle Rdb Release 7.2.1.

2.1.13 Logical RDMS$ENABLE_INDEX_COLUMN_GROUP Not Working

Bug 5614198

Users are unable to disable this feature in Oracle Rdb Release 7.2 using the VMS logical RDMS$ENABLE_INDEX_COLUMN_GROUP.

```
$define RDMS$ENABLE_INDEX_COLUMN_GROUP 0
$SQL$
attach 'file personnel';
show flags;
Alias RDB$DBHANDLE:
Flags currently set for Oracle Rdb:
   STRATEGY,PREFIX,WARN_DDL,INDEX_COLUMN_GROUP,MAX_SOLUTION
   ,MAX_RECURSION(100),DETAIL_LEVEL(1),REFINE_ESTIMATES(127)
   ,NOBITMAPPED_SCAN
```

The workaround is to use the command SQL FLAGS 'NOINDEX_COLUMN_GROUP'.

This problem has been corrected in Oracle Rdb Release 7.2.1.

2.2 SQL Errors Fixed

2.2.1 CREATE OUTLINE Not Fully Supported for MULTISCHEMA Databases

Well formed query outlines do not always work correctly within a multischema database. This is because the object name is used instead of the STORED NAME for the MODULE, RELATION and INDEX tags. The only time these references work is when the STORED NAME is the same as the name within the schema.

The following example shows the errors reported by Oracle Rdb due to the incorrectly passed names.

```
SQL> create outline S.QO_0
cont> id 'DAE28B9C6DA276E600C68C32AFF46F88'
cont> mode 0
cont> as (
cont>   query (
cont> -- Select
cont>     subquery (
cont>       TT      MODULE S.M2 0     access path sequential
cont>       )
cont>     )
cont>   )
cont> compliance optional     ;
%RDB-E-NO_META_UPDATE, metadata update failed
-RDB-E-OBSOLETE_METADA, request references metadata objects that no longer exist
-RDMS-E-MODNEXTS, module M2 does not exist in this database
SQL>
SQL> create outline S.QO
cont>     stored name is "qoQOqo"
cont> id '74263C5C965F88554C9E67744616925C'
cont> mode 0
cont> as (
cont>   query (
cont> -- For loop
cont>     subquery (
cont>       S.TTABLE 0      access path sequential
cont>       )
cont>     )
cont>   )
cont> compliance optional     ;
%RDB-E-NO_META_UPDATE, metadata update failed
-RDB-E-OBSOLETE_METADA, request references metadata objects that no longer exist
-RDMS-F-RELNOEXI, relation TTABLE does not exist in this database
SQL>
SQL> create outline S.QO_2
cont> id '21CA5C0637609367779EB2D7967FF11B'
cont> mode 0
cont> as (
cont>   query (
cont> -- For loop
cont>     subquery (
cont>       S.T 0     access path index       S.T_INDEX
cont>       )
cont>     )
cont>   )
cont> compliance optional     ;
```

```
%RDB-E-NO_META_UPDATE, metadata update failed
-RDMS-F-INDNOTEXI, index T_INDEX does not exist in this database
```

SQL was also not processing the declared local temporary (aka scratch) table name correctly. The CREATE OUTLINE syntax allows the schema (and catalog) for the table name but in the case of a temporary table within a module, the table sub-object is fully specified by the module name. SQL now restricts the name to be unqualified.

These problems have been corrected in Oracle Rdb Release 7.2.1.

2.2.2 Unexpected INV_TBL_DCL Error From CREATE MODULE in Compiled Source

If either a SQL Module Language or SQL Precompiler source file contained a CREATE MODULE statement that used DECLARE LOCAL TEMPORARY TABLE, several errors were reported. The same CREATE MODULE statement was acceptable in Interactive or Dynamic SQL.

The following example shows these errors.

```
    declare local temporary table module.T2T (f float)
                                          1
%SQL-E-INV_TBL_DCL, (1) Invalid use of declared local temporary table T2T
    select f into :x from module.T2T where module.T2T.f = 0e0;
                          1
%SQL-F-RELNOTDCL, (1) Table T2T has not been declared in module or environment
```

This restriction has been lifted with this release of Oracle Rdb. The prohibition was being incorrectly applied to CREATE MODULE statements and the restriction on DECLARE LOCAL TEMPORARY TABLE does not apply to DDL statements in a compiled module.

This problem has been corrected in Oracle Rdb Release 7.2.1.

2.2.3 Numeric Out of Range SQLSTATE 22003 Not Returned

Bug 5208860

For a table column defined as NUMERIC, if an out-of-range value were inserted in the column, the returned SQLCODE would be -1 with a SQLSTATE of RR000 in lieu of the proper SQLCODE of -304 and SQLSTATE of 22003.

For example, consider a table defined as follows:

```
CREATE TABLE NUM1 ( NUM1C1 NUMERIC (3, 2),  NUM1C2 NUMERIC (2) );
```

The following example illustrates the old, incorrect SQLCODE and SQLSTATE returned when an out-of-range value is inserted into the table:

```
SQL> INSERT INTO NUM1 VALUES (-10, 0);
%RDB-E-VALOUTRANGE, value outside the specified precision (3) for column
"NUM1C1"
SQL> show sqlca
SQLCA:
```

```
        SQLCAID:        SQLCA           SQLCABC:        128
        SQLCODE:        -1
        SQLERRD:        [0]: 2
                        [1]: 0
                        [2]: 0
                        [3]: 0
                        [4]: 0
                        [5]: 0
        SQLWARN0:       0       SQLWARN1:       0       SQLWARN2:       0
        SQLWARN3:       0       SQLWARN4:       0       SQLWARN5:       0
        SQLWARN6:       0       SQLWARN7:       0
        SQLSTATE:       RR000
```

This INSERT now produces the following results:

```
SQL> INSERT INTO NUM1 VALUES (-10, 0);
%RDB-E-VALOUTRANGE, value outside the specified precision (3) for column
"NUM1C1"
SQL> show sqlca
SQLCA:
        SQLCAID:        SQLCA           SQLCABC:        128
        SQLCODE:        -304
        SQLERRD:        [0]: 0
                        [1]: 0
                        [2]: 0
                        [3]: 0
                        [4]: 0
                        [5]: 0
        SQLWARN0:       0       SQLWARN1:       0       SQLWARN2:       0
        SQLWARN3:       0       SQLWARN4:       0       SQLWARN5:       0
        SQLWARN6:       0       SQLWARN7:       0
        SQLSTATE:       22003
```

There is no known workaround to get the correct SQLCODE/SQLSTATE.

This problem has been corrected in Oracle Rdb Release 7.2.1.

2.2.4 TIMESTAMP Result of DATE Added to TIME Expression was Truncated

Oracle Rdb allows the addition of DATE ANSI and a TIME data type value to produce a TIMESTAMP result. SQL incorrectly assigned a fractional seconds precision to the TIMESTAMP result of zero (0). Therefore, the result was truncated by Interactive SQL. SQL now correctly assigns the fractional seconds precision of the TIME value expression to the TIMESTAMP result.

The following example shows that the fractional seconds were truncated.

```
SQL> select date ansi'2006-1-1' + time'12:12:12.12' from rdb$database;

 2006-01-01 12:12:12
1 row selected
```

This problem has been corrected in Oracle Rdb Release 7.2.1. SQL now correctly assigns the fractional seconds precision of the TIME value expression to the TIMESTAMP result.

2.2.5 Unexpected Constraint Failure from INSERT ... SELECT Statement

In prior releases of Oracle Rdb, NOT DEFERRABLE constraints were evaluated for each row inserted by the INSERT INTO ... SELECT ... FROM ... statement. In some cases, this row by row evaluation might cause the INSERT statement to fail when it was expected to succeed. The ANSI and ISO Database Language standard for SQL specifies that the INSERT statement from a SELECT is atomic and constraint evaluation should be performed after all rows are inserted.

The following example shows a constraint that fails in a case where it should have succeeded.

```
SQL> set dialect 'SQL99';
SQL>
SQL> create table TEST_A (a integer);
SQL> insert into TEST_A values (1);
1 row inserted
SQL> insert into TEST_A values (-1);
1 row inserted
SQL>
SQL> create table TEST_B
cont>     (b integer);
SQL>
SQL> insert into TEST_B select * from TEST_A;
2 rows inserted
SQL>
SQL> -- add constraint that ensures total is zero
SQL> alter table TEST_B
cont>     add constraint BB
cont>   check ((select sum (b) from TEST_B) = 0)
cont>   not deferrable;
SQL>
SQL> insert into TEST_B select * from TEST_A;
%RDB-E-INTEG_FAIL, violation of constraint BB caused operation to fail
-RDB-F-ON_DB, on database DISK1:[DATABASES]MF_PERSONNEL.RDB;1
SQL>
```

This problem has been corrected in Oracle Rdb Release 7.2.1. If the SQL language dialect SQL92, SQL99, ORACLE LEVEL1 or ORACLE LEVEL2 is set, the NOT DEFERRABLE constraint evaluation is now performed after all rows have been inserted for the INSERT ... SELECT statement.

2.2.6 SQL$MOD /LIS Displays Incorrect /ALIGN Value

Bug 5350528

In some cases, the listing produced by the SQL Module Language compiler would display an incorrect value for the /(NO)ALIGN qualifier used in the command line summary section at the end of the listing. The correct value of the qualifier was used in the compilation; the problem was restricted to a misleading listing.

For example, the following might appear in the command line summary section of the listing:

```
Command Line Summary:

    /LIST/NOALIGN/MACH TEST$SOURCE:MOD_DATETIME_ADA_4.SQLMOD
```

```
/FLOAT=D_FLOAT
/WARNING=(WARNING, DEPRECATED)
/NOFLAG_NONSTANDARD
/CONSTRAINT_MODE=DEFERRED
/NOCONNECT
/INITIALIZE_HANDLES
/NORESTRICT_INVOKER
/ALIGN_RECORDS
    ...
```

In the above example, the command line specified /NOALIGN but the value shown as being used is /ALIGN_RECORDS even though the records were actually not aligned.

There is no workaround for this problem.

This problem has been corrected in Oracle Rdb Release 7.2.1.

2.2.7 FOR UPDATE Clause was Ignored for DECLARE CURSOR

Bug 4990176

Starting with Oracle Rdb Release 7.1.2, the SELECT ... FOR UPDATE clause was considered synonymous with the UPDATE ONLY CURSOR syntax. The FOR UPDATE clause is used by various SQL dialects to imply stronger locking during the initial read of a row. For instance, a singleton SELECT statement might use FOR UPDATE to fetch the row and save the DBKEY (or ROWID) for subsequent updating by the application.

For Dynamic and Interactive SQL, the FOR UPDATE is an important feature when using cursors that update rows. In these environments, the intent to update the cursor is not known until the UPDATE ... WHERE CURRENT OF statement is encountered. The FOR UPDATE clause not only provides valuable information for the optimization of the query but also locks the rows ready for update. However, the DECLARE CURSOR syntax was ignoring the FOR UPDATE clause in all dialects except ORACLE LEVEL1 and ORACLE LEVEL2 and no strong locking was applied by Oracle Rdb.

This problem has been corrected in Oracle Rdb Release 7.2.1. All SQL dialects now accept FOR UPDATE for stronger row locking.

2.2.8 UPDATE ... WHERE CURRENT OF Assigns Incorrect Result Within IF Statement

Bug 2266270

In prior versions of Oracle Rdb, an UPDATE ... WHERE CURRENT OF executed within a conditional statement such as an IF THEN ELSE or CASE statement may assign the wrong result to the target column. This can only happen under the following conditions:

- There exists more than one UPDATE ... WHERE CURRENT OF statement, each in different branches of the IF or CASE statement.
- These UPDATE statements share a common expression.

The problem occurs because the evaluation of the common expression is performed within only one branch of the IF statement and is not visible for the other conditional branches. It is these other branches that will assign the incorrect result.

The following example shows the structure of such problem queries. The common expression in this example is A + B.

```
begin
for :x as each row of cursor y
    for select id, a, b from t_table
do
    if :x.id = 1 then
        update t_table set res = a + b + 1 where current of y;
    else
        update t_table set res = a + b where current of y;
    end if;
end for;
end;
```

This problem has been corrected in Oracle Rdb Release 7.2.1.

2.2.9 Unexpected COSI−F−BUGCHECK Error Reported by SHOW Commands

Bug 5256548

In some cases, the SHOW INDEX (PARTITION), SHOW TABLE or SHOW STORAGE MAP commands may fail with a COSI−F−BUGCHECK error. This occurs when the storage area is in a UNIFORM format area and the name is exactly 31 octets in length. The error is due to the buffer being sized too small.

The following example shows the problem.

```
SQL> show table SAMPLE_TABLE;
Information for table SAMPLE_TABLE:
...
 Partition information for index:
  Partition: (1) SYS_P00150
   Storage Area: A_VERY_LONG_STORAGE_AREA_NAME_1
%COSI-F-BUGCHECK, internal consistency failure
```

This problem has been corrected in Oracle Rdb Release 7.2.1.

2.2.10 ALTER DOMAIN Incorrectly Propagates DEFAULT Changes to Table Columns

Bug 5203032

In previous releases of Oracle Rdb, the ALTER DOMAIN ... DROP DEFAULT clause would propagate the change to all columns based on that domain. While this is correct in most cases, this action should not be performed when the column has its own DEFAULT which overrides that in the domain.

The side effect was that the DROP DEFAULT would cause the column's default value to be lost. However, commands such as SHOW TABLE (COLUMN) would still display the old value.

The following example shows this behavior. Here the domain includes a CHECK constraint to prevent NULL values being inserted. The DROP DEFAULT has left neither DEFAULT expression active and an attempt is made to insert NULL.

```
SQL> create domain D_TEST
cont>     integer
cont>     default -99
cont>     check (value is not null) not deferrable;
SQL>
SQL> create table T_TEST
cont>     (ident_column D_TEST default 1
cont>     ,subident_column D_TEST default 0
cont>     );
SQL>
SQL> insert into T_TEST default values;
1 row inserted
SQL> select * from T_TEST;
 IDENT_COLUMN   SUBIDENT_COLUMN
            1                0
1 row selected
SQL>
SQL> alter domain D_TEST
cont>     drop default;
SQL>
SQL> insert into T_TEST default values;
%RDB-E-NOT_VALID, validation on field IDENT_COLUMN caused operation to fail
SQL>
```

The correction is to redefine the column default using ALTER TABLE ... ALTER COLUMN ... DEFAULT.

This problem has been corrected in Oracle Rdb Release 7.2.1. ALTER DOMAIN will no longer propagate the DROP DEFAULT changes to any column that has an overriding DEFAULT clause at the column level.

2.2.11 Module Global Variables Limited to 64 DECLARE Statements

Bug 5346544

In previous versions of Oracle Rdb, a CREATE MODULE statement with more than 64 DECLARE statements for module global variables might bugcheck. The bugcheck summary would appear similar to the one shown below.

- %COSI-F-BUGCHECK, internal consistency failure
- Exception occurred at MEM_BUGCHECK + 00000010
- Called from RDMS$$CREATE_MODULE_INFO + 000012E4
- Called from RDMS$$CREATE_MODULE_INFO + 00000C50
- Called from RDMS$$RELEASE_DDL_VM_HNDLR + 0000130C

This problem was caused by a memory allocation error and has been corrected in Oracle Rdb Release 7.2.1. Oracle Rdb now supports a virtually unlimited number of module global variables.

2.2.12 Invalid Escape Sequence Not in SQLSTATE

Bug 5208821

The SQL standard requires that an invalid escape sequence be reported in SQLSTATE with a code of "22025". Additionally, Rdb is supposed to report this condition with a SQLCODE of −1040. In prior versions of Rdb, a SQLSTATE of "RR000" and a SQLCODE of −1 were reported instead.

For example, suppose a SQL$PRE/CC program contains the following query:

```
EXEC SQL SELECT COUNT(*) FROM CPBASE
         WHERE JUNK1 LIKE 'P%X%%X' ESCAPE 'X';
```

The escape sequence in the above query is invalid because it ends with an escape character. In prior versions, dynamic SQL would have reported the error with a very generic SQLCODE of −1 and SQLSTATE of "RR000". The correct SQLCODE of −1040 and SQLSTATE of "22025" are now reported.

There is no known workaround for this problem.

This problem has been corrected in Oracle Rdb Release 7.2.1.

2.2.13 Incomplete Drop of Partitioned Indices with DROP STORAGE AREA ... CASCADE

Bug 5620530

In prior releases of Oracle Rdb, the ALTER DATABASE ... DROP STORAGE AREA ... CASCADE statement may not correctly update partitioned indices that refer to the rows deleted from a partitioned table.

The following example shows that some index nodes still reference the old rows. This is detected by RMU Verify.

```
$ DEFINE/USER RDMS$SET_FLAGS −
    "TEST_SYSTEM,STOMAP_STATS,INDEX_STAT,INTERNAL,ITEM_LIST"
$ SQL$
alter database
    filename SAMPLE_DATABASE
    drop storage area AREA_MAIN_07 cascade;
~As: Drop Storage Area "AREA_MAIN_07" Cascade
~As: ...area referenced by index: "INDEX1"
~As: ...area referenced by map: "MAP_TABLE_001"
~As: ...purge index "INDEX2"
~As: ...update the AIP for larea=65 (table)
~As: ...update the AIP for larea=77 (index)
~H Extension (VERIFY CONSTRAINTS) Item List:
0000 (00000) RDB$K_EXT_VFYC_EXCLUDE_UNIQUE
0003 (00003) RDB$K_EXT_VFYC_TABLE_NAME "TABLE_001"
000F (00015) RDB$K_INFO_END
$
$ RMU/VERIFY/INDEX SAMPLE_DATABASE
%RMU−W−CANTFINDLAREA, cannot locate logical area 65 in area inventory page list
%RMU−E−BDLAREADY, error readying logical area with dbid 65
%RMU−I−READYDATA, ready needed for data record at 65:5:0
%RMU−I−BTRNODDBK, Dbkey of B−tree node is 89:3:0
```

```
%RMU-W-BTRVFYPRU, B-tree verification pruned at this dbkey
%RMU-I-BTRPARROO, root dbkey of b-tree partition in AREA_INDEX_07 is 89:3:0
$
```

This problem has been corrected in Oracle Rdb Release 7.2.1. The only workaround for this problem is to drop and recreate the affected indices.

2.2.14 Unexpected LENMISMAT Warnings when Using TRANSLATE ... USING Function

Bug 5629307

In prior versions Oracle Rdb, the result length of the TRANSLATE (... USING ...) function was overestimated by SQL. In some cases, this caused unexpected and erroneous warnings to be issued.

The following example shows this on a simple column. There should be no warnings from this command.

```
SQL> set character length 'characters';
SQL> create table utest (u5 char(10) character set unicode) ;
SQL> show table (column) utest
Information for table UTEST

Columns for table UTEST:
Column Name                    Data Type        Domain
-----------                    ---------        ------
U5                             CHAR(10)
         UNICODE 10 Characters,  20 Octets

SQL> insert into utest values ( translate ('A' using rdb$unicode ) ) ;
1 row inserted
SQL> insert into utest values ( translate ('AB' using rdb$unicode ) ) ;
1 row inserted
SQL> insert into utest values ( translate ('ABC' using rdb$unicode ) ) ;
1 row inserted
SQL> insert into utest values ( translate ('ABCD' using rdb$unicode ) ) ;
1 row inserted
SQL> insert into utest values ( translate ('ABCDE' using rdb$unicode ) ) ;
1 row inserted
SQL> insert into utest values ( translate ('ABCDEF' using rdb$unicode ) );
%SQL-W-LENMISMAT, Truncating right hand side string for assignment to column U5
1 row inserted
SQL> insert into utest values ( translate ('ABCDEFG' using rdb$unicode ) );
%SQL-W-LENMISMAT, Truncating right hand side string for assignment to column U5
1 row inserted
SQL> commit;
```

This problem has been corrected in Oracle Rdb Release 7.2.1. SQL now uses the octet length of the maximum size character in the character set for the estimation. While it is now less likely that TRANSLATE will issue unnecessary LENMISMAT warnings, SQL may not know the final translation of the source character string, and for some variable length character sets the warning may be justified even when the assignment succeeds without truncation.

2.2.15 Unexpected Error from UNION Containing NULL Expression

Bug 5645199

In prior versions of Oracle Rdb, a UNION operator that specified NULL in the select list of the first leg might derive an invalid data type for the common data type. This could result in garbled results (as shown in the example below) or produce an error at run time.

```
%RDB-E-ARITH_EXCEPT, truncation of a numeric value at runtime
-SQL-F-UNSDTPCVT, Unsupported data type conversion
```

The following example shows the incorrect results.

```
SQL> select null from ntab
cont> union
cont> select d1 from dtab;
 D1
 @L.oGe%.....(x/z(x/z...
 NULL
2 rows selected
```

Workarounds for this problem include: wrapping a CAST expression around NULL and specifying a data type that is compatible with the other legs of the UNION, or reversing the select expressions so that the NULL expression is processed last. The next example shows the expected result.

```
SQL> select d1 from dtab
cont> union
cont> select null from ntab;
 D1
  6-NOV-2006 16:16:52.62
 NULL
2 rows selected
```

This problem has been corrected in Oracle Rdb Release 7.2.1.

2.2.16 Inconsistent Data Type Assignment to IS NULL Expression

Bug 5484239

In prior releases of Oracle Rdb, Dynamic SQL would assign a data type to a parameter marker in an IS NULL clause based on a nearby expression. In many cases, this data type was not consistently applied.

The following example shows a Dynamic SQL application prompting for data from the user. In some cases the ? IS NULL requests an integer (input 4 and 8) and at other times it requests a char(30) input.

```
Enter statement:
create table CUSTOMERS
    (id INTEGER
    ,ORDERID INTEGER
    ,NAME CHAR(30)
    );
```

```
Enter statement:
update CUSTOMERS
set ID=?, ORDERID=?, NAME=?
where (ID= ? OR (ID IS NULL AND ? IS NULL))
  and (ORDERID= ? OR (ORDERID IS NULL AND ? IS NULL))
  and (NAME= ? OR (NAME IS NULL AND ? IS NULL));
[9 fields]
 0/ID/Integer: 12345
 1/ORDERID/Integer: 345
 2/NAME/Char(30/30): Jones
 3/ID/Integer: 12355
 4//Integer: 0
 5/ORDERID/Integer: 344
 6//Char(30/30):
 7/NAME/Char(30/30): Lee
 8//Integer: 0
Enter statement:
```

This problem has been corrected in Oracle Rdb Release 7.2.1. In this release, SQL will assign the default type (that is VARCHAR(2000)) as the data type for the expression "? IS NULL".

2.2.17 Table Synonym Not Used by Query Outlines

In prior releases of Oracle Rdb, a synonym created for a table using CREATE SYNONYM or RENAME TABLE was not recognized by the query outline at runtime. A message similar to the following would be reported when the 'STRATEGY' flag was specified with the SET FLAGS statement.

```
SQL> select a from t000 order by a;
~S: Outline "QO_A" used
~S: Outline/query mismatch; assuming T000 0 renamed to TABLE_000 0
Tables:
  0 = TABLE_000
Index only retrieval of relation 0:TABLE_000
  Index name  T000_INDEX [0:0]
0 rows selected
SQL>
```

This problem has been corrected in Oracle Rdb Release 7.2.1. Oracle Rdb now compares the result target table instead of just comparing the name referenced in the query outline with that of the table in the query.

2.2.18 Unexpected UNSDTPCVT Error During String Concatenation

Bug 5584169

When a zero length character string is concatenated with another string, SQL unexpectedly reports a UNSDTPCVT error. This is due to an attempt to apply Oracle semantics to the query. This error only occurs when the dialect is set to either ORACLE LEVEL1 or ORACLE LEVEL2.

The following example shows a failing query due to this problem.

```
SQL> set dialect 'oracle level1';
SQL> select 'test'||'' from rdb$database;
%SQL-F-UNSDTPCVT, Unsupported data type conversion
SQL>
```

This problem has been corrected in Oracle Rdb Release 7.2.1.

2.2.19 Parameter for LIKE Pattern Sized Too Small

Bug 4179408

If Dynamic SQL is processing a query that uses a parameter marker for the LIKE pattern, then it currently assumes the parameter is the same length as the source string. However, a pattern such as '%123%' is perfectly valid for use in matching against a CHAR(4) column (matching leading 123 or trailing 123) but the assumed data type causes the pattern to be truncated by Oracle Rdb and consequently not all values will be matched.

The following example shows the old behavior. All three rows should be matched by the query.

```
$ run test$tools:tester
Enter statement:
attach 'filename sql$database';
Enter statement:
create table LL (val char(4));
Enter statement:
insert into LL (val) values ('1234');
Enter statement:
insert into LL (val) values ('1235');
Enter statement:
insert into LL (val) values ('0123');
Enter statement:
select * from LL where val like ?;
[1 fields]
 0/VAL/Varchar(4/8): %123%
Enter statement:
```

The following example shows the corrected behavior and increased length of the parameter marker data type.

```
$ run test$tools:tester
Enter statement:
attach 'filename sql$database';
Enter statement:
create table LL (val char(4));
Enter statement:
insert into LL (val) values ('1234');
Enter statement:
insert into LL (val) values ('1235');
Enter statement:
insert into LL (val) values ('0123');
Enter statement:
select * from LL where val like ?;
[1 fields]
 0/VAL/Varchar(8/12): %123%
 0/VAL: 1234
 0/VAL: 1235
 0/VAL: 0123
Enter statement:
```

This problem has been corrected in Oracle Rdb Release 7.2.1. SQL now assumes that the like pattern is twice the size of the source string, or if the ESCAPE clause is present, it assumes three times the size. This should allow room for most pattern strings. If this sizing is still too small, use CAST(? AS VARCHAR(n)) to size the parameter to a more precise length.

2.3 RMU Errors Fixed

2.3.1 RMU/BACKUP/AFTER Ignores Default Filename When /EDIT_FILENAME Included

Bug 5464971

When an RMU/BACKUP/AFTER command was issued and no output filename was given and the /EDIT_FILENAME qualifier was included, the default journal filename would not be used when creating the backup file. For example:

```
$ RMU/BACKUP/AFTER/LOG -
  /EDIT_STRING=("_", VNO, "_", YEAR,MONTH,DAY_OF_MONTH) -
  MF_PERSONNEL .AIJ
.
.
.
%RMU-I-LOGCREBCK, created backup file DEV:[DIR]_0_20060829.AIJ;1
```

In the above example, the journal filename was "J1" and that name should have been used as the prefix for the backup filename but instead only the contents of the edit string were used to construct the filename.

This problem can be avoided by explicitly providing the backup output filename in the backup command.

This problem has been corrected in Oracle Rdb Release 7.2.1.

2.3.2 Possible RMU COLLECT OPTIMIZER Workload Statistics Memory Corruption

Bug 5436532

A system access violation could occur when using the RMU/COLLECT OPTIMIZER command to collect WORKLOAD statistics for an Rdb database. This problem was caused by a memory corruption problem that could happen when bits were set outside of a bitmap table used to calculate workload statistics for tables with a larger number of rows. This has been fixed and bits can no longer be set outside the bounds of the bitmap table. This problem only occurs for Oracle Rdb RMU V7.2.

The following example shows the system access violation which could occur when workload statistics were collected.

```
$ RMU/COLLECT OPTIMIZER/STATISTICS=(WORKLOAD) test_datatbase
%SYSTEM-F-ACCVIO, access violation, reason mask=04,
virtual address=000000000000000C, PC=0000000000501D2C, PS=0000001B
%RMU-F-FATALOSI, Fatal error from the Operating System Interface.
%RMU-I-BUGCHKDMP, generating bugcheck dump file
device:[directory]RMUBUGCHK.DMP
```

The partial workaround for this problem is to first use the /TABLE qualifier on the RMU/COLLECT OPTIMIZER command to see which tables cause this problem and then to use the /EXCLUDE_TABLES

qualifier to not collect workload statistics for those tables.

```
$ RMU/COLLECT OPTIMIZER/STATISTICS=(WORKLOAD)-
 /EXCLUDE_TABLE=problem_table test_database
$
```

This problem has been corrected in Oracle Rdb Release 7.2.1.

2.3.3 RMU VERIFY Memory Corruption

Bugs 5349580 and 5276155

Because of database corruption, it was possible for a counter used to allocate memory for the structures which guide the database verify to be too small. This can happen if the corruption causes queries of the system tables to return inconsistent data. One cause of this could be corruption of Area Bitmap Pages for uniform areas.

When other queries were made to the database to fill entries in the verify structures for tables, indexes, segmented strings, etc., entries put in the verify structures for the various database objects could go beyond the memory allocated for the structures. This caused memory corruption which could result in a system error and an RMU/VERIFY bugcheck.

The database corruption causing inconsistent data to be returned from the system tables cannot be repaired by the verify. But now checks are made to prevent an overrun of the memory size allocated for the verify structures and a warning message will be output that not all of the database objects will be verified because of a problem accessing the system tables but that the verify will continue. Since this is a case where queries to the database cannot be trusted and there is no way to tell what is causing this inconsistency at the point where it happens at the start of the verify, this gives a chance for the verify to continue so it can show the problem.

The following example shows a case where an RMU/VERIFY of a database encountered corruption which caused a memory overrun when loading the verify structures. This caused memory corruption which resulted in repeated system reserved operand faults. The verify tried to continue but when it could not load the structures needed to go on with the verify, the verify aborted and output a bugcheck dump because of an unexpected system error.

```
$ RMU/VERIFY/ALL device:[directory]database.rdb
%RMU-I-BGNROOVER, beginning root verification
%RMU-I-ENDROOVER, completed root verification
%SYSTEM-W-ROPRAND, reserved operand fault at PC=000000000032E178,
PS=0000001B
%RMU-E-ERRRDBIND, error accessing RDB$INDICES relation
%RMU-I-PARTLVFY,  continuing partial verification
%RMU-F-ABORTVER,  fatal error encountered; aborting verification
%SYSTEM-F-ROPRAND, reserved operand fault at PC=000000000032E178,
PS=0000001B
%RMU-F-FATALOSI, Fatal error from the Operating System Interface.
%RMU-I-BUGCHKDMP, generating bugcheck dump file
 device:[directory]RMUBUGCHK.DMP;
%RMU-F-FTL_VER, Fatal error for VERIFY operation at 21-JUN-2006
08:03:57.63
```

The following example shows the corrected behavior. The memory overrun is prevented and a warning message is output for the particular database object type stating that there was a problem loading some of the information to completely verify all objects of that type. The verify continues so that any diagnostics it returns

can help identify the problem. In the case of the Area Bit Map page corruption, an RMU/REPAIR can be executed to fix the corruption problem.

```
$ RMU/VERIFY/ALL device:[directory]database.rdb
%RMU-W-NOTALLDAT, Not all data for database TABLES can be loaded from
system tables - verify continuing.
%RMU-W-ABMBITERR, inconsistency between spam page 9475371 and bit 918
in area bitmap in larea 1 page 6
%RMU-W-ABMBITERR, inconsistency between spam page 9490631 and bit 932
in area bitmap in larea 1 page 6
%RMU-W-ABMBITERR, inconsistency between spam page 9491721 and bit 933
in area bitmap in larea 1 page 6
%RMU-E-BADABMPAG,       error verifying ABM pages
$ RMU/REPAIR/ABM/AREA=UNIFORM_AREA device:[directory]DATABASE.RDB
%RMU-I-FULBACREQ, A full backup of this database should be performed
 after RMU REPAIR
$ RMU/VERIFY/ALL device:[directory]DATABASE.RDB
```

This problem has been corrected in Oracle Rdb Release 7.2.1.

2.3.4 Unexpected DLMNOTFND When Leading Characters of Suffix Appear in Data

Bug 4245634

In prior versions of Oracle Rdb, RMU Load would report an error if the data included the leading character from the SUFFIX. This occurred with FORMAT=DELIMITED when a SUFFIX option was specified. For example, the name "SMITH, ANDREW", in the sample data below, is followed by a "/" which is also the leading character of the SUFFIX defined on the RMU Load command line.

```
/:/key3/:/,/:/SMITH, ANDREW//:/,/:/Z/:/&END&
```

This causes the RMU Load to fail as shown below.

```
$ RMU/LOAD/RECORD_DEFINITION=(-
    FILE=NAMES_TABLE.RRD,-
    FORMAT=DELIMITED_TEXT, -
    PREFIX="/:/",-
    SUFFIX="/:/",-
    TERMINATOR="&END&",-
    NULL="")-
    LOAD_TEST NAMES_TABLE NAMES.DAT
%RMU-F-DLMNOTFND, Separator (,) not found for column 2 of row 1 in the input.
%RMU-I-DATRECREAD,  3 data records read from input file.
%RMU-I-DATRECSTO,   0 data records stored.
%RMU-I-DATRECREJ,   0 data records written to exception file.
%RMU-F-FTL_LOAD, Fatal error for LOAD operation at 14-JUL-2006 12:49:18.79
```

This problem has been corrected in Oracle Rdb Release 7.2.1. RMU Load now correctly scans ahead for the correct SUFFIX.

2.3.5 Unexpected Failure of RMU Load When the NULL String Appears With Column Data

Bug 4865227

In prior releases of Oracle Rdb, the RMU Load command would fail if it detected the NULL string within column data. This occurs with FORM=DELIMITED and specifying the NULL option to the RECORD_DEFINITION qualifier. The following example shows a simple case.

```
$ RMU/UNLOAD/RECORD=(FILE=T1.RRD,FORMAT=DELIMITED,-
        SEPARATOR="|",PREFIX="",SUFFIX="",NULL="***") -
        SAMPLE_DB T1 T1.DAT
%RMU-I-DATRECUNL,   1 data records unloaded.
$ TYPE T1.DAT
abcde|N***N     |z
$ RMU/LOAD/RECORD=(FILE=T1.RRD,FORMAT=DELIMITED,-
        SEPARATOR="|",PREFIX="",SUFFIX="",NULL="***") -
        SAMPLE_DB T1 T1.DAT
  DEFINE FIELD C1 DATATYPE IS TEXT SIZE IS 5.
  DEFINE FIELD C2 DATATYPE IS TEXT SIZE IS 10.
  DEFINE FIELD C3 DATATYPE IS TEXT SIZE IS 1.
  DEFINE RECORD T1.
     C1 .
     C2 .
     C3 .
  END T1 RECORD.
%RMU-F-UNEXPDELIM, Unexpected delimiter encountered (***) in row 1 of input
%RMU-I-DATRECREAD,  1 data records read from input file.
%RMU-I-DATRECSTO,   0 data records stored.
%RMU-F-FTL_LOAD, Fatal error for LOAD operation at 14-JUL-2006 00:57:40.42
```

RMU Load should not have reported an error in this case as the data is not ambiguous.

This problem has been corrected in Oracle Rdb Release 7.2.1.

2.3.6 Unexpected INVALID_BLR Error Reported For RMU Load

Bug 4040175

In prior releases of Oracle Rdb, the RMU Load command would accept a record definition file that contained the data types PACKED DECIMAL, UNSIGNED NUMERIC and LEFT SIGNED NUMERIC. This would lead to errors similar to this example:

```
$ rmu/load abc temp_table /rec=(format=text,file=z.rrd) z.dat
%RDB-E-INVALID_BLR, request BLR is incorrect at offset 8
%RMU-I-DATRECREAD,  0 data records read from input file.
%RMU-I-DATRECSTO,   0 data records stored.
%RMU-F-FTL_LOAD, Fatal error for LOAD operation at 13-JUL-2006 20:50:23.81
```

These types are not supported by the Oracle Rdb server and should not be allowed by RMU Load. This release will correctly diagnose the use of these types.

```
%RMU-F-UNSSUPDAT, Unsupported data type: 16
%RMU-I-DATRECSTO,   0 data records stored.
%RMU-F-FTL_LOAD, Fatal error for LOAD operation at 13-JUL-2006 20:50:40.17
```

This problem has been corrected in Oracle Rdb Release 7.2.1.

2.3.7 RMU /COPY, /MOVE and /RESTORE Could Corrupt ABM Pages

Bug 5276155

For RMU COPY, MOVE and RESTORE, the Area Bit Map pages for uniform storage areas could be incorrectly set if there was a chain of multiple ABM bitmap pages for a logical area and the previous ABM page of the chain had NO bits set but subsequent ABM pages in the chain had bits set. This could happen if there were enough SPAM pages in the uniform storage area so that not all SPAM PAGES could be represented by the bit map in a single ABM page so multiple ABM pages, each with a bitmap, were required to represent the SPAM pages for each logical area. If one of the ABM pages (but not the last) for a logical area had NO bits set, then any ABM pages following it that had bits set before the RMU MOVE, COPY or RESTORE would have all bits cleared after the RMU COPY, MOVE or RESTORE. If a bit is not set for an ABM page for a logical area, then Rdb will not retrieve any database data pages controlled by that SPAM page. Note that this can only happen for the case described and only for uniform storage areas. This will be detected by RMU/VERIFY and can be fixed by RMU/REPAIR/ABM/AREAS. This problem has been fixed and now the bits in the ABM bitmaps will be correctly set in this case.

The following example shows a case where an RMU/COPY of a database causes this problem. The problem is then detected by an RMU/VERIFY and fixed by an RMU/REPAIR.

```
$ RMU/COPY/NOLOG/DIRECTORY=device:[directory] DATABASE.RDB
$ RMU/VERIFY/AREAS/LAREAS device:[directory]DATABASE.RDB
%RMU-W-ABMBITERR, inconsistency between spam page 8650241 and bit 161 in
area bitmap in larea 113 page 8650819
%RMU-W-ABMBITERR, inconsistency between spam page 8651331 and bit 162 in
area bitmap in larea 113 page 8650819
%RMU-W-ABMBITERR, inconsistency between spam page 8652421 and bit 163 in
area bitmap in larea 113 page 8650819
%RMU-W-ABMBITERR, inconsistency between spam page 8653511 and bit 164 in
area bitmap in larea 113 page 8650819
%RMU-W-ABMBITERR, inconsistency between spam page 8654601 and bit 165 in
area bitmap in larea 113 page 8650819
%RMU-W-ABMBITERR, inconsistency between spam page 8655691 and bit 166 in
area bitmap in larea 113 page 8650819
%RMU-W-ABMBITERR, inconsistency between spam page 8656781 and bit 167 in
area bitmap in larea 113 page 8650819
%RMU-E-BADABMPAG,       error verifying ABM pages
$ RMU/REPAIR/ABM/AREA=UNIFORM_AREA device:[directory]DATABASE.RDB
%RMU-I-FULBACREQ, A full backup of this database should be performed
 after RMU REPAIR
$ RMU/VERIFY/AREAS/LAREAS device:[directory]DATABASE.RDB
$
```

This problem has been corrected in Oracle Rdb Release 7.2.1.

2.3.8 RMU Unload May Truncate REAL Data When Delimited or Text Format Used

Bug 4297346

In prior releases of Oracle Rdb, an RMU Unload of a virtual column might in some rare cases result in the data being truncated.

This could occur if the computation was a table COMPUTED BY or view column that calculated the result using LEAST, GREATEST, CASE, NULLIF, COALESCE, NVL, NVL2, or DECODE expression. For this problem to occur, the computation must include one expression that resulted in REAL (F Floating) and another that resulted in SMALLINT. Unfortunately, Oracle Rdb was promoting the result to DOUBLE PRECISION (G Floating) prior to converting the value to a string value. When the value was delivered to RMU Unload, the target string, which was sized correctly for a REAL string value, was too small for the resulting DOUBLE PRECISION string and the result was truncated.

The workaround for this problem is to include an explicit CAST(... AS REAL) in the COMPUTED BY or view column definition.

The following output shows the truncation of the column:

```
003537||| 9.7500000E+01|||975||| 9.750000000000
```

The expected result would include the exponent.

```
003537||| 9.7500000E+01|||975||| 9.750000000000000E+001
```

This problem has been corrected in Oracle Rdb Release 7.2.1.

2.3.9 Incomplete Multischema Database Support in RMU Extract

Bugs 5558122, 5568079, and 5568040

In prior releases of Oracle Rdb, RMU Extract had incomplete support for multischema databases. With this release, the following corrections have been made to RMU Extract.

- Extracted query outlines did not output the names of tables, indices or modules correctly. Only the STORED NAME was used which caused the generated script to fail.
- Extracted views included the STORED NAME IS clause but the name may not have been delimited when it contained lowercase characters or different character set values. This also caused errors when executing the generated script.
- Most other objects did not include the STORED NAME IS clause at all and so there was possibly conflict with names generated by SQL for tables and any view definitions that were subsequently extracted.

These problems have been corrected in Oracle Rdb Release 7.2.1.

2.3.10 RMU Extract Item=Protections Did Not Consistently Extract Protections

Bug 5225643

In previous versions of Oracle Rdb, the RMU Extract Item=Protections output for a table might include the unused OPERATOR privilege for the table or omit the REFERENCES privilege for a module, function or procedure. These errors are harmless but the second error could prevent any routine created by the generate script being a target for synonym.

This problem has been corrected in Oracle Rdb Release 7.2.1. RMU now consistently outputs the protections for all objects.

2.3.11 RMU/CONVERT Bugchecks in PIO$LOCK_PAGE When Statistics Disabled

If statistics were disabled while executing an RMU/CONVERT command, the RMU utility would bugcheck with a stack footprint similar to the following:

```
***** Exception at 0000000000719708 : RMU72\PIO$DEMOTE_PAGE + 000001A8
%SYSTEM-F-ACCVIO, access violation, reason mask=00, virtual
address=0000000000000000, PC=0000000000719708, PS=0000001B
Saved PC = 0000000000722E14 : RMU72\PIOUTL$EMPTY_ONE_BUFFER + 000002B4
Saved PC = 000000000071D654 : RMU72\PIOFETCH$WITHIN_DB_HNDLR + 00000134
Saved PC = FFFFFFFF81104EC8 : Image LIBOTS + 00008EC8
Saved PC = FFFFFFFF800A693C : symbol not found
***** Exception at 000000000071C020 : RMU72\PIO$LOCK_PAGE + 00000320
Saved PC = 000000000071E114 : RMU72\PIOFETCH$WITHIN_DB + 00000924
Saved PC = 000000000071B444 : RMU72\PIOFETCH$FETCH + 000002E4
Saved PC = 000000000071A4E4 : RMU72\PIO$FETCH + 000008F4
```

The same problem might also occur when an implied conversion was done by restoring a backup that was made with a prior version of Oracle Rdb.

This problem can be demonstrated with the following commands:

```
$ @SYS$LIBRARY:RDB$SETVER 71
Current PROCESS Oracle Rdb environment is version V7.1-441 (MULTIVERSION)
Current PROCESS SQL environment is version V7.1-441 (MULTIVERSION)
Current PROCESS Rdb/Dispatch environment is version V7.1-441 (MULTIVERSION)
$ MCR SQL$71
SQL> CREATE DATABASE FILENAME TEST;
SQL> EXIT
$
$ DEFINE RDM$BIND_STATS_ENABLED 0
$
$ @SYS$LIBRARY:RDB$SETVER 72
Current PROCESS Oracle Rdb environment is version V7.2-010 (MULTIVERSION)
Current PROCESS SQL environment is version V7.2-010 (MULTIVERSION)
Current PROCESS Rdb/Dispatch environment is version V7.2-010 (MULTIVERSION)
$ RMU/CONVERT/NOCONFIRM TEST
%RMU-I-RMUTXT_000, Executing RMU for Oracle Rdb V7.2-010
%RMU-I-LOGCONVRT, database root converted to current structure level
%RMU-I-BUGCHKDMP, generating bugcheck dump file dev:[dir]RMUBUGCHK.DMP;
```

This problem can be avoided by deassigning the RDM$BIND_STATS_ENABLED logical prior to executing the RMU/CONVERT command.

This problem has been corrected in Oracle Rdb Release 7.2.1.

2.3.12 RMU/RESTORE/AREA/ONLINE Was Unnecessarily Locking RDB$SYSTEM for PROTECTED READ

Bug 5203722

RMU/RESTORE/AREA/ONLINE was locking the RDB$SYSTEM area in Protected Read mode even though RDB$SYSTEM was not one of the storage areas being restored. RMU/RESTORE/AREA/ONLINE does have to access the system area in order to read and possibly change the Area Inventory Pages if they are inconsistent. The Area Inventory Pages are located in the system area. However, RMU was unnecessarily locking the system area even when only mixed format storage areas, which do not have AIP pages referencing them, were being restored.

Now the locking and accessing of the system area AIP pages has been changed for an online by area restore. If all areas being restored are of MIXED format, the system area is not locked if it is not one of the areas being restored. If any uniform areas are being restored online, then the system area will be readied in Concurrent Write mode instead of Protected Read mode to provide more concurrency during the restore. For uniform areas, the AIP pages need to be accessed to help reconstruct the Area Bit Map and Space Management Pages and possibly changed if they are inconsistent.

Note that any of the areas actually being restored which are named in the restore command, including the system area, will continue to be locked for Exclusive Update.

The following example shows the online by area restore commands affected by these changes. If the restore references only mixed areas, there is no locking of the system area. If the restore references any uniform areas, there will now be Concurrent Write locking of the system area where there was formerly Protected Read locking of the system area. If the system area is itself being restored, it will continue to be locked for exclusive update.

```
$ RMU/RESTORE/ONLINE/AREA/NOLOG/DIR=DEVICE:[DIRECTORY] -
         DEVICE:[DIRECTORY]DATABASE.RBF MIXED_1_AREA, MIXED_2_AREA
$ RMU/RESTORE/ONLINE/AREA/NOLOG/DIR=DEVICE:[DIRECTORY] -
         DEVICE:[DIRECTORY]DATABASE.RBF MIXED_1_AREA, MIXED_2_AREA, -
                        UNIF_1_AREA, UNIF_2_AREA
$ RMU/RESTORE/ONLINE/AREA/NOLOG/DIR=DEVICE:[DIRECTORY] -
         DEVICE:[DIRECTORY]DATABASE.RBF MIXED_1_AREA, MIXED_2_AREA,-
                        UNIF_1_AREA, UNIF_2_AREA, RDB$SYSTEM
```

This problem has been corrected in Oracle Rdb Release 7.2.1.

2.3.13 Failure of RMU /BACKUP of Database With Very Large Storage Area Count and Small Specified /BLOCK_SIZE Value

Bug 5376038

Previously, databases with a very large number of storage areas and a specified small value for /BLOCK_SIZE might cause RMU /BACKUP to fail. An internal buffer could overflow the output file block size and would either write a record that could not be read during recovery or could corrupt memory and cause an ACCVIO bugcheck during the backup. For some versions, the buffer overflow would be detected at run time and the RMU /BACKUP operation would exit with a bugcheck dump with a "footprint" similar to the following:

```
$ RMU/BACKUP FOO.RDB FOO.RBF /BLOCK_SIZE=2048
***** Exception at 003E95F0 : RMUBCK$BACKUP_SUMMARY + 00000270
%COSI-F-BUGCHECK, internal consistency failure
Saved PC = 003E8584 : RMUBCK$BF_BACKUP_THREAD + 00000494
Saved PC = 008302FC : RMUIO$TERMINATE_THREAD + 00000040
```

```
Saved PC = 008304B8 : RMUIO$FIREWALL + 00000040
Saved PC = 00830478 : RMUIO$FIREWALL + 00000000
Saved PC = 003A18E4 : RMU_DISPATCH + 00000434
Saved PC = 003A1008 : RMU_STARTUP + 000004E8
Saved PC = 001E002C : RMU$MAIN + 0000002C
```

This problem has been corrected in Oracle Rdb Release 7.2.1. RMU /BACKUP now correctly adjusts the backup block size as needed to accommodate the number of database storage areas and page sizes.

2.3.14 RMU Extract Reports BAD_CODE Error for BITSTRING Function

In prior releases of Oracle Rdb, RMU Extract would generate a BAD_CODE error when trying to extract a BITSTRING function nested within a CASE, ABS, COALESCE, DECODE, NULLIF, NVL, or NVL2 function. The following example shows the reported error.

```
%RMU-F-BLRINV, internal error - BLR string 83 for . is invalid
-RDMS-E-BAD_CODE, corruption in the query string
%RMU-F-FTL_RMU, Fatal error for RMU operation at 10-JUL-2006 03:27:48.68
```

This problem has been corrected in Oracle Rdb Release 7.2.1.

2.3.15 Incorrect SQL Syntax Generated for Views Containing UNION and GROUP BY Clauses

Bugs 5666305 and 5672460

In prior releases, RMU Extract would incorrectly extract view definitions by including an asterisk "*" following the select value list when a UNION included a branch containing a GROUP BY clause.

This example shows the original view definition.

```
SQL> create view SAMPLE_VIEW (a, b, c)
cont>     as
cont>     select last_name, first_name, middle_initial
cont>     from candidates
cont>     group by last_name, first_name, middle_initial
cont> union
cont>     select last_name, first_name, middle_initial
cont>     from employees
cont>     group by last_name, first_name, middle_initial
cont> ;
```

This is the output from RMU Extract showing the incorrect syntax.

```
   .
   .
   .
create view SAMPLE_VIEW
    (A,
     B,
     C) as
    select C3.LAST_NAME, C3.FIRST_NAME, C3.MIDDLE_INITIAL
        * from CANDIDATES C3
```

```
        group by C3.LAST_NAME, C3.FIRST_NAME, C3.MIDDLE_INITIAL
    union
    select C5.LAST_NAME, C5.FIRST_NAME, C5.MIDDLE_INITIAL
        * from EMPLOYEES C5
        group by C5.LAST_NAME, C5.FIRST_NAME, C5.MIDDLE_INITIAL;
.
.
.
```

This problem has been corrected in Oracle Rdb Release 7.2.1.

2.4 LogMiner Errors Fixed

2.4.1 RMU /UNLOAD /AFTER_JOURNAL Incorrect NULL Bit Setting

Bug 5256671

In prior Oracle Rdb releases, certain modifications and structures of database table definitions could result in the LogMiner improperly processing the null column information.

The following example demonstrates one possible cause of incorrect NULL processing. Note that in the extracted records, incorrect columns are indicated as NULL.

```
$ DEFINE /NOLOG SQL$DATABASE FOO
$ SQL$
    CREATE DATA FILE SQL$DATABASE
        NUMBER OF BUFFERS 100 NUMBER OF CLUSTER NODES 1;
    DISCONNECT ALL;
    ALTER DATA FILE SQL$DATABASE
        JOURNAL ENA (FAST COMMIT ENA) ADD JOURNAL J1 FILE J1;
    CREATE TABLE T1 (
        I1 INT,I2 INT,I3 INT,I4 INT,I5 INT,I6 INT,
        I7 INT,I8 INT,I9 INT,I10 INT,I11 INT,I12 INT);
    CREATE STORAGE MAP M1 FOR T1 DISABLE COMPRESSION;
    COMMIT;
    ALTER TABLE T1 DROP COLUMN I5;
    COMMIT;
    ALTER TABLE T1
        ADD COLUMN I13 INT
        ADD COLUMN I14 INT
        ADD COLUMN I15 INT
        ADD COLUMN I16 INT
        ADD COLUMN I17 INT;
    COMMIT;
    ALTER TABLE T1 DROP COLUMN I17;
    ALTER TABLE T1 ADD COLUMN I18 INT;
    COMMIT;
    ALTER TABLE T1 DROP COLUMN I18;
    COMMIT;

    DISCONNECT ALL;
    EXIT;
$ RMU/SET LOGMINER/ENABLE/NOLOG SQL$DATABASE
$ RMU/BACKUP/AFTER/NOLOG SQL$DATABASE NLA0:BAR
$ RMU/BACKUP/NOLOG/NOCRC/NOCHECKSUM SQL$DATABASE NLA0:BAR
$ SQL$
    INSERT INTO T1 VALUES (1,2,3,4,6,7,8,9,10,11,12,13,14,15,16);
    INSERT INTO T1 VALUES (NULL,2,3,4,6,7,8,9,10,11,12,13,14,15,16);
    INSERT INTO T1 VALUES (1,NULL,3,4,6,7,8,9,10,11,12,13,14,15,16);
    INSERT INTO T1 VALUES (1,2,NULL,4,6,7,8,9,10,11,12,13,14,15,16);
    INSERT INTO T1 VALUES (1,2,3,NULL,6,7,8,9,10,11,12,13,14,15,16);
    INSERT INTO T1 VALUES (1,2,3,4,NULL,7,8,9,10,11,12,13,14,15,16);
    INSERT INTO T1 VALUES (1,2,3,4,6,NULL,8,9,10,11,12,13,14,15,16);
    INSERT INTO T1 VALUES (1,2,3,4,6,7,NULL,9,10,11,12,13,14,15,16);
    INSERT INTO T1 VALUES (1,2,3,4,6,7,8,NULL,10,11,12,13,14,15,16);
    INSERT INTO T1 VALUES (1,2,3,4,6,7,8,9,NULL,11,12,13,14,15,16);
```

```
    INSERT INTO T1 VALUES (1,2,3,4,6,7,8,9,10,NULL,12,13,14,15,16);
    INSERT INTO T1 VALUES (1,2,3,4,6,7,8,9,10,11,NULL,13,14,15,16);
    INSERT INTO T1 VALUES (1,2,3,4,6,7,8,9,10,11,12,NULL,14,15,16);
    INSERT INTO T1 VALUES (1,2,3,4,6,7,8,9,10,11,12,13,NULL,15,16);
    INSERT INTO T1 VALUES (1,2,3,4,6,7,8,9,10,11,12,13,14,NULL,16);
    INSERT INTO T1 VALUES (1,2,3,4,6,7,8,9,10,11,12,13,14,15,NULL);
    COMMIT;
$ RMU/BACKUP/AFTER/NOLOG SQL$DATABASE B1
$ RMU/UNLOAD/AFTER/NOLOG/FORMAT=DUMP SQL$DATABASE B1 -
    /TABLE=(NAME=T1,OUTPUT=T1.DAT)
$ SEARCH T1.DAT NULL
I13                             : NULL
I9                              : NULL
I13                             : NULL
I10                             : NULL
I13                             : NULL
I11                             : NULL
I13                             : NULL
I12                             : NULL
I13                             : NULL
I13                             : NULL
I14                             : NULL
I13                             : NULL
I15                             : NULL
I13                             : NULL
I16                             : NULL
I13                             : NULL
I13                             : NULL
I13                             : NULL
I13                             : NULL
I13                             : NULL
I13                             : NULL
I13                             : NULL
I13                             : NULL
$ EXIT
```

This problem has been corrected in Oracle Rdb Release 7.2.1. The LogMiner now correctly tracks the length of internal null information structures based on the table definition.

2.4.2 RMU /UNLOAD /AFTER_JOURNAL AERCP_LEN Field Incorrect In Text Format

Bug 5617814

Previously, when using the TEXT or DELIMITED_TEXT output format, the AERCP_LEN field of the commit record content would be incorrect.

For example, when using delimited text format, the following information might be returned in a commit record:

```
"C",
"",
    .
    .
    .
"16",
"7169",
"00000000-0000-0000-0000-000000000000",
```

```
"000000000000038F000000000000038F0000016000000000000001C01"
```

Here, the field value "16" is the RM_TID_LEN and the "7169" value is the incorrect AERCP_LEN field.

This problem has been corrected in Oracle Rdb Release 7.2.1. The correct value for the AERCP_LEN field is now returned. Note that the AERCP_LEN value is 28 and this represents the unformatted binary length of the AERCP structure and not the length of the text formatted field.

2.5 Row Cache Errors Fixed

2.5.1 RMU/RECOVER of Journaled Row Cache Changes Corrupts Database

Bug 5469750

If a database had row cache parameters changed, and the database was restored and recovered, the resulting database would be corrupt. Sometimes the RMU/RECOVER process would fail as well, and occasionally the Oracle Rdb monitor process would fail.

The following script demonstrates the problem.

```
$
$ ! Create a database with a few basic caches.
$
$ SQL$
  CREATE DATABASE FILENAME TEST
  NUMBER OF CLUSTER NODES 1
  RESERVE 4 STORAGE AREAS
  RESERVE 3 JOURNALS
  RESERVE 3 CACHE SLOTS
  ROW CACHE IS ENABLED
  CREATE STORAGE AREA RDB$SYSTEM FILENAME TEST
  CREATE STORAGE AREA TEST_A1 FILENAME TEST_A1
  CREATE STORAGE AREA TEST_A2 FILENAME TEST_A2
  CREATE STORAGE AREA TEST_A3 FILENAME TEST_A3
  CREATE CACHE TEST_A1 CACHE SIZE 100 ROWS ROW LENGTH 100 BYTES
  CREATE CACHE TEST_A2 CACHE SIZE 200 ROWS ROW LENGTH 200 BYTES
  CREATE CACHE TEST_A3 CACHE SIZE 300 ROWS ROW LENGTH 300 BYTES;
  DISCONNECT ALL;

  ALTER DATABASE FILENAME TEST
    ADD JOURNAL TEST_J1 FILENAME SYS$DISK:[]TEST_J1.AIJ
    ADD JOURNAL TEST_J2 FILENAME SYS$DISK:[]TEST_J2.AIJ
    ADD JOURNAL TEST_J3 FILENAME SYS$DISK:[]TEST_J3.AIJ
    JOURNAL IS ENABLED (FAST COMMIT ENABLED);
%RDMS-W-DOFULLBCK, full database backup should be done to ensure future recovery
  EXIT;
$
$ ! Save away the original cache configuration.
$
$ RMU/BACKUP/NOLOG TEST TEST
$ RMU/DUMP/HEADER=BRIEF/OUTPUT=BEFORE.TXT TEST
$
$ ! Alter a cache parameter.
$
$ SQL$
  ALTER DATABASE FILENAME TEST ALTER CACHE TEST_A2 ROW LENGTH 400 BYTES;
  DISCONNECT ALL;

-- Delete the database

  DROP DATABASE FILENAME TEST;
  EXIT;
$
```

```
$ ! Restore the database.  RMU will automatically recover the journals.
$
$ RMU/RESTORE/NOCDD/NOLOG TEST
%RMU-I-AIJRSTAVL, 3 after-image journals available for use
%RMU-I-AIJRSTMOD, 1 after-image journal marked as "modified"
%RMU-I-AIJISON, after-image journaling has been enabled
%RMU-W-DOFULLBCK, full database backup should be done to ensure future recovery
%RMU-I-LOGRECDB, recovering database file DEV:[DIR]TEST.RDB;1
%RMU-I-AIJAUTOREC, starting automatic after-image journal recovery
%RMU-I-AIJONEDONE, AIJ file sequence 0 roll-forward operations completed
%RMU-I-AIJONEDONE, AIJ file sequence 1 roll-forward operations completed
%RMU-W-NOTRANAPP, no transactions in this journal were applied
%RMU-I-AIJALLDONE, after-image journal roll-forward operations completed
%RMU-I-AIJSUCCES, database recovery completed successfully
%RMU-I-AIJFNLSEQ, to start another AIJ file recovery, the sequence number
needed will be 2
$ RMU/DUMP/HEADER=BRIEF/OUTPUT=AFTER.TXT TEST
$
$ ! Compare the original database with the recovered database.
$ ! In this example, instead of the cache row length being changed
$ ! to 400 the number of database buffers is changed to 400.
$
$ DIFFERENCE BEFORE.TXT AFTER.TXT
.
.
.
************
File DEV:[DIR]BEFORE.TXT;1
   35         - Default user buffer count is 20
   36         - Default recovery buffer count is 20
   37         - Global buffers are disabled
******
File DEV:[DIR]AFTER.TXT;1
   35         - Default user buffer count is 400
   36         - Default recovery buffer count is 400 (stored as 20)
   37         - Global buffers are disabled
************
```

Depending on what row cache parameters were changed, various failures may occur in the RMU/RECOVER operation or in the database monitor. In the reported problem, RMU/RECOVER would fail with the following exception:

```
***** Exception at 007E35BC : PIO$FETCH + 000003EC
%SYSTEM-F-ACCVIO, access violation, reason mask=00,
virtual address=0000000000000000, PC=00000000007E35BC, PS=0000001B
```

Also, the database monitor failed with the following exception:

```
**** Exception at hhhhhhhh : MON$DELETE_UNREFERENCED_GBL + 00000DAC
%SYSTEM-F-ACCVIO, access violation, virtual address=0000000000414000
```

To avoid this problem, do a full database and journal backup after altering any row cache parameters. If this problem is encountered, it is possible to recover the restored database up until the point in the journal that contains the row cache changes. That is, using the /UNTIL qualifier, recover the journals up to the point in time that the row cache changes were made.

This problem has been corrected in Oracle Rdb Release 7.2.1.

2.5.2 Recovered Database Corrupt When ROW SNAPSHOT IS ENABLED

When the snapshots in cache feature was enabled, it was possible for after−image journal (AIJ) entries to be logged with incorrect transaction sequence numbers (TSNs). This could result in a corrupt database if the journal was used to recover a restored database. The problem would occur if an error happened during an update statement and the transaction was later committed. For example, a constraint failure or lock timeout followed by a COMMIT could cause incorrect journal entries to be made.

This problem was introduced in Oracle Rdb Releases 7.1.4.4 and 7.2.0.2.

This problem can be avoided by setting all caches to ROW SNAPSHOT IS DISABLED.

This problem has been corrected in Oracle Rdb Release 7.2.1.

2.6 RMU Show Statistics Errors Fixed

2.6.1 RMU/SHOW STATISTICS Hot Standby Statistics State Display Field

Bug 5396571

Previously, when using the TCP/IP network transport with the Hot Standby feature, the RMU /SHOW STATISTICS "Hot Standby Statistics" display "State:" field could overwrite the "UserSync:" heading as in the following example:

```
Node: HSVMS  (1/1/16)  Oracle Rdb V7.1-441 Perf. Monitor 18-JUL-2006 06:17:59.97
Rate: 3.00 Seconds           Hot Standby Statistics         Elapsed: 00:07:28.63
Page: 1 of 1  $1$DGA113:[MWILLEMS.HS.HS1.MASTER]MF_PERSONNEL.RDB;1  Mode: Online
------------------------------------------------------------------------------
State: TCP/IP:72     rSync: Cold Current.Msg: 1       Cl Mstr.AIJ: 1:2
LagTime: 00:00:00 AutoSync: Cold Stalled.Msg: none     1 Stby.AIJ: 1:2
Stby.DB: HSVMS1::$1$DGA20:[MWILLEMS.HS.HS1.STANDBY]STANDBY_MF_PERSONNEL
```

The line starting with "State:" partly overwrites "UserSync:".

This problem has been corrected in Oracle Rdb Release 7.2.1.

2.6.2 RMU /SHOW STATISTICS Defined Logicals List Incomplete

Bug 5600122

Previously, it was likely that the RMU /SHOW STATISTICS Defined Logicals display did not properly list all logicals when the display was set to "Full" mode. This problem was caused by an incorrect calculation of the number of logical names possible.

This problem has been corrected in Oracle Rdb Release 7.2.1. The full list of logical names is correctly displayed.

2.6.3 Rdb Executive Sort and Temporary Work File Statistics

Bug 5617519

Previously, there was no reliable way to determine the number of sorting operations nor temporary work file operations within the Rdb executive at a database-wide level.

New statistics are available to help understand the number and type of sorting operations and temporary work file operations within the Rdb executive. These statistics are available on the "Rdb Executive Statistics" screen of the RMU /SHOW STATISTICS utility:

- records sorted − The number of data items passed in to a sort routine (note that the elements being sorted may not be actual database rows).
- quick−sorts − Sort operations that were handled entirely within an internally buffered quick sort routine. These are generally sorts of smaller cardinalities of simple sort keys.
- sort32 sorts − Sort operations that were handled by the internal SORT32 interface.
- workfile write IO − Write IO operations to SORT32 on−disk work files.
- workfile read IO − Read IO operations from SORT32 on−disk work files.
- temp file create − Creation of temporary relation work files.
- delete − Deletion of temporary relation work files.
- record put − Data written using RMS to temporary relation work files.
- record get − Data read using RMS from temporary relation work files.
- truncate − Rewind and truncate of temporary relation work files using RMS.
- record position − Rewind or backspace operation of temporary relation work files using RMS.

Example Rdb Executive Statistics display with sort and temporary work file statistics:

```
Node: LUCAS (1/1/1)    Oracle Rdb X7.2−00 Perf. Monitor  26−OCT−2006 22:41:29.41
Rate: 3.00 Seconds          Rdb Executive Statistics        Elapsed: 00:02:50.14
Page: 1 of 1           $1$DGA203:[HSEC]MF_PERSONNEL.RDB;1           Mode: Online
------------------------------------------------------------------------------
statistic.........      rate.per.second..... total....... average......
name..............      max..... cur..... avg....... count....... per.trans....
queries compiled              16        0        0.7           82          16.4
index scans                   84        0        3.8          430          86.0
 index only                    0        0        0.0            0           0.0
 index full                   21        0        1.0          112          22.4
dynamic optimizer             10        0        0.3           34           6.8
 one abandoned                 0        0        0.0            1           0.2
 all abandoned                 0        0        0.0            0           0.0
records sorted                37        0        3.0          339          67.8
 quick−sorts                   0        0        0.0            3           0.6
 sort32 sorts                  0        0        0.0            4           0.8
  workfile write IO            0        0        0.0            0           0.0
  workfile read IO             0        0        0.0            0           0.0
temp file create              63        0        2.7          301          60.2
          delete              63        0        2.7          301          60.2
        truncate               0        0        0.0            0           0.0
      record put             246        0       11.3         1268         253.6
      record get             267        0       11.3         1268         253.6
 record position              63        0        2.7          301          60.2
```

Chapter 3
Software Errors Fixed in Oracle Rdb Release 7.2.0.2

This chapter describes software errors that are fixed by Oracle Rdb Release 7.2.0.2.

3.1 Software Errors Fixed That Apply to All Interfaces

3.1.1 AUTOMATIC AS Columns Changed to NULL Expression by DROP ... CASCADE

Bug 5194374

In prior releases of Oracle Rdb, the convention was to convert the computed expression to return NULL when a COMPUTED BY column was invalidated because of a dropped table, view, module, function or procedure. This allowed queries against the table to succeed and report an UNKNOWN result.

However, there are several problems with this convention in the current release:

- AUTOMATIC AS columns are also affected and this causes NULL to be written as column values. AUTOMATIC AS columns should not be changed to generate NULL.
  If the AUTOMATIC UPDATE AS column includes a constraint (such as NOT NULL), then UPDATE on that table will fail.
- If the COMPUTED BY column was based on a sequence, no such setting to NULL was performed making this convention inconsistent.
- Generally when a sequence, table, or routine is dropped, the metadata is in a transitory state. The database administrator will recreate the missing objects soon, probably in the same transaction. However, this permanent change to the metadata forces the database administrator to add extra ALTER TABLE ... ALTER COLUMN statements to the script to redefine the AUTOMATIC column.

With this release of Oracle Rdb, the check for an invalid COMPUTED BY is deferred until query compile. The original definition is no longer changed and will be used once the missing definition is replaced. AUTOMATIC AS columns are no longer affected by this change, however, an error will be reported if an AUTOMATIC AS column is incomplete when an INSERT or UPDATE statement is attempted.

All metadata changes which cause the error OBSOLETE_METADATA will now be detected at query compile time and the NULL expression substituted for COMPUTED BY column values. This includes missing tables, views, functions, procedures, sequences and synonyms.

This problem has been corrected in Oracle Rdb Release 7.2.0.2.

3.1.2 Possible DBR or RCS Failure After Database Convert to V7.2

Bug 5099828

It is possible that when a database is converted to Rdb V7.2 format, an internal data structure in the Rdb root file may be left incorrectly initialized. If certain bits are set in this data structure, it is possible that a DBR or RCS process may fail in the routine LOGFIL$GET_FILNAM with a bugcheck exception "footprint" similar to the following:

```
COSI-F-BUGCHECK, internal consistency failure
Exception occurred at RDMDBR72\LOGFIL$GET_FILNAM + 00000108
Called from RDMDBR72\DBR$MAIN + 00000B7C
Called from symbol not found
Called from RDMDBR72\DBR$MAIN + 00000B7C
Running image RDMDBR72.EXE
```

This problem has been corrected in Oracle Rdb Release 7.2.0.2. The database conversion now correctly initializes the data structure.

3.1.3 Reduced CPU Usage and Improved Performance of CRC and Checksum Calculations

Several performance enhancements to CRC and Checksum calculations have been implemented in this release of Oracle Rdb. CRC and Checksum calculations are used during database page reading and writing and for RMU backup and restore operations. The enhancements include:

- More aggressive CPU Cache prefetching
- Promotion of memory fetches from longword to quadword
- More aggressive loop unrolling
- Streamlined instruction sequences

3.1.4 Inconsistent Snapshot Results Using COMMIT TO JOURNAL

Bug 5024150

When the COMMIT TO JOURNAL OPTIMIZATION feature was enabled, it was possible for processes executing READ ONLY transactions to get inconsistent results when reading from the database. The problem could be encountered when the following sequence of events occurred:

1. A READ ONLY transaction starts.
2. A READ WRITE transaction starts.
3. The READ WRITE transaction deletes a row from a table and then commits.
4. The READ WRITE transaction inserts a new row in the table reusing the space freed by the previous delete, and commits.
5. The READ ONLY transaction attempts to read the row just deleted/inserted from the database.

In the above sequence of events, the READ ONLY transaction would conclude that the row was deleted instead of using the contents of the row that were current at the time that the READ ONLY transaction started.

Note that utilities that use READ ONLY transactions, such as online backups or verifies, could also encounter the problem. Online backups would backup an empty row instead of the old contents. Online verifies would typically return errors like the following:

```
%RMU-W-BADIDXREL, Index I1 either points to a non-existent record or
                  has multiple pointers to a record in table T1.
                  The logical dbkey in the index is 57:6:0.
```

This problem can be avoided by disabling the COMMIT TO JOURNAL OPTIMIZATION option.

This problem has been corrected in Oracle Rdb Release 7.2.0.2.

3.1.5 Wrong Result From UNION Query With OR Predicates

Bug 5092217

The following UNION query with OR predicates containing constants returns the wrong result (should return 1 row).

```
SET FLAGS 'STRA,DETAIL';
SELECT V.ACODE, V.DATA FROM
(SELECT DISTINCT G.ACODE, G.DATA, M.MID  FROM
    GAREA  G,
    RSEC   S,
    MSEC   M,
    BDAY   C
 WHERE  S.ACODE = G.ACODE AND
        S.SID   = M.SID AND
        M.MID   = C.MID
 UNION
 SELECT DISTINCT G.ACODE, G.DATA, M.MID  FROM
    GAREA  G,
    OBASK  B,
    MBASK  M,
    BDAY   C
 WHERE  G.ACODE = B.ACODE AND
        B.BID   = M.BID AND
        M.MID   = C.MID
 ) AS V (ACODE, DATA, MID)
 WHERE V.MID= 1 AND
      ( '  ' ='' OR '  ' = V.ACODE ) ;
Tables:
  0 = GAREA
  1 = RSEC
  2 = MSEC
  3 = BDAY
  4 = GAREA
  5 = BASKET
  6 = MBASK
  7 = MBASK
  8 = BDAY
Conjunct: (<mapped field> = 1 AND ('  ' = <mapped field>))
Merge of 1 entries
  Merge block entry 1
  Reduce: <mapped field>, <mapped field>, <mapped field>
  Sort: <mapped field>(a), <mapped field>(a), <mapped field>(a)
  Merge of 2 entries
    Merge block entry 1
    Reduce: 0.ACODE, 0.DATA, 2.MID
    Sort: 0.ACODE(a), 0.DATA(a), 2.MID(a)
    Cross block of 3 entries
      Cross block entry 1
        Conjunct: 2.MID = 3.MID
        Match
          Outer loop      (zig-zag)
            Index only retrieval of relation 3:BDAY
              Index name  BDAY_PK [0:0]
```

```
          Inner loop      (zig-zag)
            Conjunct: 2.MID = 1
            Index only retrieval of relation 2:MSEC
              Index name  MSEC_PK [1:1]
                Keys: <mapped field> = 1
      Cross block entry 2
        Get     Retrieval by index of relation 1:RSEC
          Index name  RSEC_PK [1:1]     Direct lookup
            Keys: 1.SID = 2.SID
      Cross block entry 3
        Get     Retrieval by index of relation 0:GAREA
          Index name  GAREA_PK [1:1]         Direct lookup
            Keys: 1.ACODE = 0.ACODE
            Bool: ('  ' = '') OR ('  ' = 0.ACODE)
    Merge block entry 2
    Reduce: 4.ACODE, 4.DATA, 7.MID
    Sort: 4.ACODE(a), 4.DATA(a), 7.MID(a)
    Cross block of 4 entries
      Cross block entry 1
        Conjunct: 7.MID = 1
        Get     Retrieval sequentially of relation 7:MBASK
      Cross block entry 2
        Index only retrieval of relation 8:BDAY
          Index name  BDAY_PK [1:1]       Direct lookup
            Keys: 7.MID = 8.MID
      Cross block entry 3
        Cross block of 2 entries
          Cross block entry 1
            Get     Retrieval by index of relation 5:BASKET
              Index name  BASKET_PK [1:1]    Direct lookup
                Keys: 5.BID = 7.BID
          Cross block entry 2
            Conjunct: 5.BID = 6.BID
            Get     Retrieval sequentially of relation 6:MBASK
      Cross block entry 4
        Get     Retrieval by index of relation 4:GAREA
          Index name  GAREA_PK [1:1]         Direct lookup
            Keys: 4.ACODE = 5.ACODE
            Bool: <error: common keyonly boolean no predicates>
0 rows selected
```

The problem is obvious from the following debug trace.

```
      Cross block entry 4
        Get     Retrieval by index of relation 4:GAREA
          Index name  GAREA_PK [1:1]         Direct lookup
            Keys: 4.ACODE = 5.ACODE
            Bool: <error: common keyonly boolean no predicates>
```

The error in the above boolean should represent the following boolean predicates:

```
            Bool: ('  ' = '') OR ('  ' = 0.ACODE)
```

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

1. The main query selects from the derived table of a union query merged from two subqueries.
2. The WHERE clause of the main query contains filter predicates of AND and OR applying the columns of the derived table.

3. The left operand of the OR predicate is a constant expression and the right operand is an expression referencing one of the columns of the derived table.

As a workaround, the query works if the operands of the OR predicate are swapped even though the debug trace is still showing the error in the boolean.

```
SELECT V.ACODE, V.DATA FROM
(SELECT DISTINCT G.ACODE, G.DATA, M.MID  FROM
    GAREA  G,
    RSEC   S,
    MSEC   M,
    BDAY   C
 WHERE  S.ACODE = G.ACODE AND
        S.SID   = M.SID AND
        M.MID   = C.MID
 UNION
 SELECT DISTINCT G.ACODE, G.DATA, M.MID  FROM
    GAREA  G,
    OBASK  B,
    MBASK  M,
    BDAY   C
 WHERE  G.ACODE = B.ACODE AND
        B.BID   = M.BID AND
        M.MID   = C.MID
 ) AS V (ACODE, DATA, MID)
 WHERE V.MID= 1 AND
      ( '  ' = V.ACODE          -- this operand is swapped with
        OR
        '  ' =''        ) ;     -- this constant operand
Tables:
  0 = GAREA
  1 = RSEC
  2 = MSEC
  3 = BDAY
  4 = GAREA
  5 = BASKET
  6 = MBASK
  7 = MBASK
  8 = BDAY
Conjunct: (<mapped field> = 1 AND ('  ' = <mapped field>))
Merge of 1 entries
  Merge block entry 1
  Reduce: <mapped field>, <mapped field>, <mapped field>
  Sort: <mapped field>(a), <mapped field>(a), <mapped field>(a)
  Merge of 2 entries
    Merge block entry 1
    Reduce: 0.ACODE, 0.DATA, 2.MID
    Sort: 0.ACODE(a), 0.DATA(a), 2.MID(a)
    Cross block of 3 entries
      Cross block entry 1
        Conjunct: 2.MID = 3.MID
        Match
          Outer loop      (zig-zag)
            Index only retrieval of relation 3:BDAY
              Index name  BDAY_PK [0:0]
          Inner loop      (zig-zag)
            Conjunct: 2.MID = 1
            Index only retrieval of relation 2:MSEC
              Index name  MSEC_PK [1:1]
                Keys: <mapped field> = 1
      Cross block entry 2
```

```
        Get     Retrieval by index of relation 1:RSEC
          Index name  RSEC_PK [1:1]     Direct lookup
            Keys: 1.SID = 2.SID
      Cross block entry 3
        Get     Retrieval by index of relation 0:GAREA
          Index name  GAREA_PK [1:1]         Direct lookup
            Keys: 1.ACODE = 0.ACODE
            Bool: ('  ' = '') OR ('  ' = 0.ACODE)
    Merge block entry 2
    Reduce: 4.ACODE, 4.DATA, 7.MID
    Sort: 4.ACODE(a), 4.DATA(a), 7.MID(a)
    Cross block of 4 entries
      Cross block entry 1
        Conjunct: 7.MID = 1
        Get     Retrieval sequentially of relation 7:MBASK
      Cross block entry 2
        Index only retrieval of relation 8:BDAY
          Index name  BDAY_PK [1:1]       Direct lookup
            Keys: 7.MID = 8.MID
      Cross block entry 3
        Cross block of 2 entries
          Cross block entry 1
            Get     Retrieval by index of relation 5:BASKET
              Index name  BASKET_PK [1:1]    Direct lookup
                Keys: 5.BID = 7.BID
          Cross block entry 2
            Conjunct: 5.BID = 6.BID
            Get     Retrieval sequentially of relation 6:MBASK
      Cross block entry 4
        Get     Retrieval by index of relation 4:GAREA
          Index name  GAREA_PK [1:1]         Direct lookup
            Keys: 4.ACODE = 5.ACODE
            Bool: <error: common keyonly boolean no predicates>
 ACODE      DATA
 CH         Foo
1 row selected
```

This problem has been corrected in Oracle Rdb Release 7.2.0.2.

3.1.6 Wrong Result from Left Outer Join Query with CAST in the WHERE Predicate

Bug 5123448

The customer reports that no records are returned from a query of nested left outer join operations with CAST functions embedded in the WHERE predicates.

After simplifying the query, the problem is reproduced with a much simpler query using one single left outer join operation with only two rows of data.

```
create table T1 (C1  CHAR (16),
                 C2  DATE VMS,
                 C3  DATE VMS );

insert into t1 value
 ('1K641294','01000','19-MAR-2006 19:05:45.00','19-MAR-2006 20:30:48.00');
insert into t1 value
 ('1K641294','01100','19-MAR-2006 20:31:54.00','19-MAR-2006 20:31:54.00');
```

```
SET FLAGS 'stra,detail';
Select * from
   (select D2.C1, D2.C2, D2.C3
    from
      (select D1.C1, D1.C2, D1.C3
      from
        (select  * from T1) as D1
      group by D1.C1, D1.C3, D1.C2) as D2
      left outer join
      T1 as D4
        on D2.C3 = D4.C3
    ) as D3
WHERE
  cast(D3.C2 as timestamp)  < cast(D3.C3 as timestamp)
    ;
Tables:
  0 = T1
  1 = T1
Merge of 1 entries
  Merge block entry 1
  Conjunct: CAST (0.C2 AS TIMESTAMP) < CAST (0.C3 AS TIMESTAMP) <= See Note
  Cross block of 2 entries        (Left Outer Join)
    Cross block entry 1
      Merge of 1 entries
        Merge block entry 1
        Reduce: 0.C1, 0.C3, 0.C2
        Sort: 0.C1(a), 0.C3(a), 0.C2(a)
        Merge of 1 entries
          Merge block entry 1
          Conjunct: CAST (0.C2 AS TIMESTAMP) < CAST (0.C3 AS TIMESTAMP)
          Get     Retrieval sequentially of relation 0:T1
    Cross block entry 2
      Conjunct: 0.C3 = 1.C3
      Get     Retrieval sequentially of relation 1:T1
0 rows selected
```

The query works if the CAST functions are removed from the operands of the predicate, as in the following example.

```
Select * from
   (select D2.C1, D2.C2, D2.C3
    from
      (select D1.C1, D1.C2, D1.C3
      from
        (select  * from T1) as D1
      group by D1.C1, D1.C3, D1.C2
      ) as D2
      left outer join T1 as D4
       on D2.C3 = D4.C3
    ) as D3
WHERE
  D3.C2 < D3.C3
! cast(D3.C2 as timestamp)  < cast(D3.C3 as timestamp)
    ;
Tables:
  0 = T1
  1 = T1
Conjunct: 0.C2 < 0.C3                   <= See Note
Merge of 1 entries
  Merge block entry 1
```

```
  Cross block of 2 entries        (Left Outer Join)
    Cross block entry 1
      Merge of 1 entries
        Merge block entry 1
        Reduce: 0.C1, 0.C3, 0.C2
        Sort: 0.C1(a), 0.C3(a), 0.C2(a)
        Merge of 1 entries
          Merge block entry 1
          Conjunct: 0.C2 < 0.C3
          Get     Retrieval sequentially of relation 0:T1
    Cross block entry 2
      Conjunct: 0.C3 = 1.C3
      Get     Retrieval sequentially of relation 1:T1
 C1                  C2                          C3
 1K641294            19-MAR-2006 19:05:45.00   19-MAR-2006 20:30:48.00
1 row selected
```

Note:: Notice the differences in the detail strategy output between the bad and good queries.

In the case of the bad query, the conjunct:

```
"CAST (0.C2 AS TIMESTAMP) < CAST (0.C3 AS TIMESTAMP)"
```

is located outside the Cross block but under the Merge block.

In the case of the good query, the conjunct:

```
"0.C2 < 0.C3"
```

is located at the outermost of the whole query strategy.

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

1. The main SELECT query selects from a main derived table with a WHERE clause that contains CAST functions.
2. The main derived table is the output from a left outer join query which joins a nested derived table on table T1 with GROUP BY and the table T1 itself.
3. The operands of the WHERE predicate reference the columns from the GROUP BY subquery of the left outer join.

This problem has been corrected in Oracle Rdb Release 7.2.0.2.

3.1.7 Divide−by−zero Arithmetic Error in Sampled Selectivity

Bug 4055309

The following query generates a divide−by−zero arithmetic error when the sampled selectivity is enabled by setting the SQL flag to 'selectivity(2)'.

```
set flags 'selectivity(2)';
select T2.num_dest
  from T2 T2, T1 T1
  where T1.num_dest = T2.num_dest
```

```
    AND T2.del_log = 'N'
    AND T1.del_log = 'N'
    AND T1.num_ach = 13535723;
%RDB-E-ARITH_EXCEPT, truncation of a numeric value at runtime
-SYSTEM-F-HPARITH, high performance arithmetic trap, Imask=00000000, Fmask=00000
001, summary=04, PC=000000000XXXXXX, PS=0000000B
-SYSTEM-F-FLTDIV, arithmetic trap, floating/decimal divide by zero at PC=0000000
00XXXXXXX, PS=0000000B
```

The query works if the sampled selectivity is disabled, as in the following example.

```
set flags 'noselectivity';
select T2.num_dest
  from T2 T2, T1 A
  where T1.num_dest = T2.num_dest
    AND T2.del_log = 'N'
    AND T1.del_log = 'N'
    AND T1.num_ach = 13535723;
Tables:
  0 = T2
  1 = T1
Cross block of 2 entries
  Cross block entry 1
    Conjunct: 1.DEL_LOG = 'N'
    Get     Retrieval by index of relation 1:T1
      Index name  T1_NDX [1:1]      Direct lookup
        Keys: 1.NUM_ACH = 13535723
  Cross block entry 2
    Leaf#01 FFirst 0:T2 Card=1977689
      Bool: (1.NUM_DEST = 0.NUM_DEST) AND (0.DEL_LOG = 'N')
      BgrNdx1 T2_NDX [1:1] Fan=30
        Keys: 1.NUM_DEST = 0.NUM_DEST
0 rows selected
```

This problem has been corrected in Oracle Rdb Release 7.2.0.2.

3.2 SQL Errors Fixed

3.2.1 Unexpected SQL Bugcheck When Running OCI Services for Rdb

Bug 5195499

In prior releases of Oracle Rdb, applications using the OCI Services for Rdb might generate a bugcheck dump. The summary would be similar to the following:

- Itanium OpenVMS 8.2−1
- Oracle Rdb Server 7.2.0.1.0
- SQL*Net for Rdb
- SQL−F−BUGCHK, There has been a fatal error.
- Exception occurred at SQL$SHR72\SQL$$CREATE_STMT + 000000D0
- Called from SQL$SHR72\SQL$$WALK_DATATYPE + 00002090
- Called from SQL$SHR72\SQL$$SEMANTICS + 00000860
- Called from SQL$SHR72\SQL$PREPARE + 00000760
- Running image RDB$NATCONN72.EXE

This problem occurred when the application attempted to use the SAVEPOINT command, which is not supported in Oracle Rdb. This problem has been corrected in Oracle Rdb Release 7.2.0.2.

3.2.2 SHOW MODULE Incorrectly Displays Domain Name in Header

Bug 5192211

When the variable used in a module references a domain, the SHOW MODULE command gives the wrong module name in the "Routines in module" header. Instead it displays the last domain displayed in the variables list.

The following example shows the incorrect output.

```
SQL> SHOW MODULE MOD_X
Information for module MOD_X

 Header:
 mod_x language sql
 declare :var_i dom_z
 No description found
 Module ID is: 7

Variables for module MOD_X:

Variable  Name                  Data Type        Domain or Type
--------------                  ---------        --------------
VAR_I                           INTEGER          DOM_Z

 Routines in module DOM_Z:
```

```
     XXX
SQL>
```

This problem has been corrected in Oracle Rdb Release 7.2.0.2.

3.2.3 ALTER INDEX ... BUILD ALL PARTITIONS May Bugcheck

Bug 5092374

In prior releases of Oracle Rdb, an ALTER INDEX ... BUILD ALL PARTITIONS for a HASHED index that was executed in a new database session might result in a bugcheck dump. The following example shows the error:

```
SQL> attach 'f testing_database';
SQL> alter index xtest_idx build all partitions;
%RDMS-I-BUGCHKDMP, generating bugcheck dump file DISK:[USER]RDSBUGCHK.DMP;
%RDMS-I-BUGCHKDMP, generating bugcheck dump file DISK:[USER]SQLBUGCHK.DMP;
%SYSTEM-F-ACCVIO, access violation, reason mask=00, virtual
address=000000000000 00D8, PC=0000000000236A0C, PS=0000001B
```

A workaround for this problem is to name each partition and use a BUILD PARTITION clause for each partition. Complete this sequence with an ALTER INDEX ... MAINTENANCE IS ENABLED statement. For example:

```
SQL> attach 'f testing_database';
SQL> alter index xtest_idx build partition a;
%RDB-W-META_WARN, metadata successfully updated with the reported warning
-RDMS-W-IDXNOBLDPEND, index not in build pending state - build ignored
SQL> alter index xtest_idx build partition b;
%RDB-W-META_WARN, metadata successfully updated with the reported warning
-RDMS-W-IDXNOBLDPEND, index not in build pending state - build ignored
SQL> alter index xtest_idx build partition c;
%RDB-W-META_WARN, metadata successfully updated with the reported warning
-RDMS-W-IDXNOBLDPEND, index not in build pending state - build ignored
SQL> alter index xtest_idx maintenance is enabled;
SQL> commit;
```

This problem has been corrected in Oracle Rdb Release 7.2.0.2. This problem does not affect SORTED or SORTED RANKED indices.

3.2.4 After ROLLBACK in an XA 2PC, 1PC Transactions Fail

Bug 2457148

If a SQL Module Language or Precompiled SQL application submits a ROLLBACK statement while an explicit distributed transaction is active, it will receive an "%RDB-E-ROLBCKNAL, rollback not allowed" error. This is normal and expected since a SQL statement may not be used to end an explicit distributed transaction. However, after the distributed transaction was rolled back properly, SQL would fail when a non-distributed transaction (one phase commit or 1PC) was attempted and issue the following error: "%RDB-E-DISTABORT, distributed transaction was aborted". This problem has now been corrected and 1PC and 2PC transactions can be intermixed so long as there is only one transaction active at any given point.

The following sequence of events shows an example of how the problem was encountered:

- The application calls the XA TM and starts or joins a transaction.
- The application issues SQL statements to Rdb by calling a SQL Module Language routine.
- The application issues a SQL ROLLBACK statement to Rdb by calling a SQL Module Language routine. This call returns a SQLCODE of −1 and a message code of %RDB−E−DISTABORT, which is expected since a SQL COMMIT or ROLLBACK is not allowed during this form of distributed transaction.
- The application calls the XA TM to end and rollback the transaction.
- The application issues SQL statements to Rdb by calling a SQL Module Language routine. At this point a new, 1PC transaction should be implicitly started. But instead, the call would return a SQLCODE of −1 and a message of %RDB−E−DISTABORT. This is the problem which has been corrected.

The workaround to this problem is to not submit a ROLLBACK statement during a 2PC but instead to properly end the 2PC via the distributed transaction interface.

This problem has been corrected in Oracle Rdb Release 7.2.0.2.

3.2.5 ALTER SEQUENCE ... RESTART WITH Clause Sometimes Ignored

Bug 5102072

In prior releases of Oracle Rdb, the ALTER SEQUENCE ... RESTART WITH clause was ignored if the value provided was the same as the current initial value. This initial value may be set by CREATE SEQUENCE using the START WITH clause, inherited from the MINVALUE (or MAXVALUE) when START WITH is not provided, or modified by a prior ALTER SEQUENCE ... RESTART WITH clause.

The following example shows the problem.

```
SQL> set flags 'trace';
SQL>
SQL> create sequence test_order start with 1;
SQL> show sequence test_order;
     TEST_ORDER
 Sequence Id: 1
 Initial Value: 1
 Minimum Value: 1
 Maximum Value: 9223372036854775806
 Next Sequence Value: 1
 Increment by: 1
 Cache Size: 20
 No Order
 No Cycle
 No Randomize
 Wait
SQL>
SQL> begin
cont> trace test_order.nextval;
cont> trace test_order.nextval;
cont> end;
~Xt: 1
~Xt: 2
```

```
SQL>
SQL> alter sequence test_order restart with 1;
SQL>
SQL> begin
cont> trace test_order.nextval;
cont> trace test_order.nextval;
cont> end;
~Xt: 3
~Xt: 4
SQL>
```

The final block of output, after the RESTART WITH, should be 1 and 2.

This problem has been corrected in Oracle Rdb Release 7.2.0.2. Rdb no longer ignores the RESTART WITH clause in this case.

3.2.6 DROP SYNONYM Requires DBADM Privilege for Synonyms Created by RENAME Statement

When RENAME is used to change the name of database objects, the original name is used to create a synonym referencing the new object name. This synonym is a vital part of the RENAME command and allows existing constraint, trigger, view, routine and other definitions to execute using the old name. However, the DROP SYNONYM command no longer has dependencies recorded in the database for this name and therefore allows both a RESTRICT and a CASCADE drop of these synonyms.

This may result in unusable definitions, say for a view, that requires this name. If the synonym has been dropped by mistake, then it can be recreated using the CREATE SYNONYM statement.

With this release of Oracle Rdb, Release 7.2.0.2, the DROP SYNONYM statement will additionally require the user to have the DBADM (administrator) privilege on the database if the synonym was created by RENAME. The following example shows the reported error when DBADM is required.

```
SQL> rename table EMPLOYEES to EMPLOYEE_RECORDS;
SQL> drop synonym EMPLOYEES restrict;
%RDB-E-NO_META_UPDATE, metadata update failed
-RDB-E-NO_PRIV, privilege denied by database facility
```

3.2.7 Extra Columns Added by CREATE TABLE ... LIKE Corrupt Table Data

Bug 5211731

The CREATE TABLE ... LIKE statement generates an incorrect table definition when extra columns are added, as shown in the following example. The EMPLOYEE_ID should be '00164'.

```
SQL> create table ONE_EMPLOYEE
cont>     like EMPLOYEES (tag varchar(1));
SQL> insert into ONE_EMPLOYEE
cont>     select *,'~' from EMPLOYEES where employee_id = '00164';
1 row inserted
SQL> select employee_id from ONE_EMPLOYEE;
```

```
 EMPLOYEE_ID
 ..~64
1 row selected
```

The error occurs because the offset in the row for the extra columns is not correctly set and therefore data from several columns is mapped to the same part of the record. The workaround for this problem is to add the extra columns using ALTER ... ADD COLUMN. This problem has been corrected in Oracle Rdb Release 7.2.0.2.

3.3 RMU Errors Fixed

3.3.1 RMU Online Backup Bugchecks When Snapshot in Row Cache Enabled

When using the Snapshots in Row Cache feature, it was possible for an online RMU backup operation to bugcheck with a footprint similar to:

```
Exception at 0075DBA4 : PIOFETCH$WITHIN_DB + 000005E4
%RMU-F-CANTREADDBS, error reading pages 959:4294967295-4294967295
-RMU-F-BADPAGNUM, page 4294967295 is out of valid range (1:394) for
physical area 959
```

This type of problem is generally caused by a line in the live storage area being not visible to the backup operation and there being no visible line in the snapshot storage areas as well.

In the case of using the Snapshots in Row Cache feature, two potential causes of this problem have been corrected:

- During rollback of a modification to the database, the prior copy of a line was written back to the live storage area but the snapshots for that line were not written back from cache to the snapshot storage area.
- During writing of snapshots from cache to the snapshot storage area, it was possible for the write operation to terminate before all required snapshots had been written.

These problems have been corrected in Oracle Rdb Release 7.2.0.2.

3.3.2 RMU/BACKUP/DISK_FILE/PARALLEL Problem Assigning Parameters to Executor Processes

Bug 5058214

For RMU Parallel Backup to multiple disk files, there was a problem assigning disk directories to executor processes when the number of directories exceeded the number of executors. The directory parameters were not distributed evenly among the executors: some executors might be assigned too many directories while other executors might not be assigned any directories and were dropped from the PLAN file.

The method of assigning directory parameters to worker executor processes has been changed so that the directories are distributed evenly among the executors and all executors are assigned directories and are used in the backup. Note that in cases where the number of executors is greater than the number of disk directory parameters, the extra executors will continue to be dropped with a warning message since it makes no sense to create executors with nothing to do.

The following example shows a case where an RMU/BACKUP/DISK_FILE/PARALLEL command specifies 5 disk directory parameters to be assigned among 4 executor worker processes. This problem caused only 3 executor processes to be used in the backup.

```
$ RMU/BACKUP/DISK_FILE/PARALLEL=EXECUTORS=4 -
```

```
 TEST_DATABASE -
 DISK:[DIRECTORY1]TEST_DATABASE.RBF, -
 DISK:[DIRECTORY2],-
 DISK:[DIRECTORY3],-
 DISK:[DIRECTORY4],-
 DISK:[DIRECTORY5]
```

A workaround for this problem is to specify an equal number of executor processes and disk directory parameters.

```
$ RMU/BACKUP/DISK_FILE/PARALLEL=EXECUTORS=5 -
 TEST_DATABASE -
 DISK:[DIRECTORY1]TEST_DATABASE.RBF, -
 DISK:[DIRECTORY2],-
 DISK:[DIRECTORY3],-
 DISK:[DIRECTORY4],-
 DISK:[DIRECTORY5]
```

These problems have been corrected in Oracle Rdb Release 7.2.0.2.

3.3.3 RMU/LOAD/AUDIT System Access Violation for Empty ACE Field

Bug 5078983

A system access violation occurred on an RMU/LOAD/AUDIT command since Oracle Rdb RMU did not handle a case where an access control entry field in a record from the system audit file might be empty. A negative length was calculated which was interpreted as an extremely large positive length value for the access control entry which caused an access violation when allocated memory was exceeded.

The following example shows the system access violation occurring while loading the VMS system audit file records for the Rdb database TEST_DATABASE into the MFP_AUDIT table in the AUDIT_DB Rdb database.

```
$ RMU/LOAD/AUDIT=DATABASE_FILE=TEST_DATABASE AUDIT_DB MFP_AUDIT -
 sysmgr$audit:XXXXX_060201-060301.AUDIT;1
 %SYSTEM-F-ACCVIO, access violation, reason mask=00, virtual
 address=0000000000C7E000, PC=00000000004DEC4C, PS=0000001B
 %RMU-I-BUGCHKDMP, generating bugcheck dump file
 DISK:[DIRECTORY]RMUBUGCHK.DMP;
 %RMU-I-DATRECREAD,  28881 data records read from input file.
 %RMU-I-DATRECSTO,   0 data records stored.
 %RMU-F-FTL_LOAD, Fatal error for LOAD operation at  1-MAR-2006 10:42:07.38
```

A workaround for this problem is to specify /COMMIT_EVERY=1 for the first load that fails with the system access violation. This ensures that all records up to the record that gives the access violation will be loaded. Then repeat the load specifying /SKIP=28881 to continue loading with the system audit record after the problem record which is the last record read before the access violation occurs. This will load all audit records except the problem record so it is not a complete workaround.

```
$ RMU/LOAD/AUDIT=DATABASE_FILE=TEST_DATABASE/COMMIT_EVERY=1 -
 AUDIT_DB MFP_AUDIT -
 sysmgr$audit:XXXXX_060201-060301.AUDIT;1
 %SYSTEM-F-ACCVIO, access violation, reason mask=00, virtual
 address=0000000000C7E000, PC=00000000004DEC4C, PS=0000001B
```

```
 %RMU-I-BUGCHKDMP, generating bugcheck dump file
 DISK:[DIRECTORY]RMUBUGCHK.DMP;
 %RMU-I-DATRECREAD,  28881 data records read from input file.
 %RMU-I-DATRECSTO,   595 data records stored.
 %RMU-F-FTL_LOAD, Fatal error for LOAD operation at  1-MAR-2006 10:42:07.38
$ RMU/LOAD/AUDIT=DATABASE_FILE=TEST_DATABASE/SKIP=28881 -
 AUDIT_DB MFP_AUDIT -
 sysmgr$audit:XXXXX_060201-060301.AUDIT;1
```

This problem has been corrected in Oracle Rdb Release 7.2.0.2.

3.3.4 Some Functions Missing from RMU Extract Script

Bug 5148011

In prior releases of Oracle Rdb, RMU Extract would ignore modules starting with the RDB$ prefix. This meant that the two modules created by SQL_FUNCTIONS script: RDB$ORACLE_SQLFUNC_CHAR and RDB$ORACLE_SQLFUNC_OCTET, would be missing from the created database.

In this case the following routines would not be defined: ADD_MONTHS, ASCII, INITCAP, INSTR, LAST_DAY, LPAD, LTRIM, MONTHS_BETWEEN, NEW_TIME, NEXT_DAY, RDB$GMT_OFFSET, REPLACE, RPAD, RTRIM, SUBSTR, INSTRB, SUBSTRB.

This problem has been corrected in Oracle Rdb Release 7.2.0.2. RMU Extract no longer ignores these modules.

3.3.5 Infrequent and Intermittent Access Violation During RMU/COPY

Bug 4904628

An infrequent and intermittent access violation during RMU/COPY that only occurred on the VMS Integrity Platform has been fixed. The problem occurred in the thread scheduling code and caused the stack to get corrupted and the thread context to be lost. This caused a system access violation. This problem only occurred in a narrow window where two or more threads could intefere with each other.

The following example shows the system access violation occurring while copying a database on the VMS Integrity platform.

```
RMU/COPY/LOG/ONLINE TEST_DATABASE/DIR=DEVICE:[DIRECTORY]-
 /ROOT=DEVICE:[DIRECTORY] /SNAPSHOT=(ALLOCATION=10)
%SYSTEM-F-ACCVIO, access violation, reason mask=00,
 virtual address=00000000000000B0, PC=FFFFFFFF800C3B00, PS=0000001B
```

This problem has been corrected in Oracle Rdb Release 7.2.0.2.

3.3.6 Invalid RMU/BACKUP %RMU-W-BADPTLARE Warning if TRUNCATE TABLE

An invalid %RMU−W−BADPTLARE warning message could be output by Oracle Rdb RMU/BACKUP for database pages that were deleted by a SQL TRUNCATE TABLE statement. The backup completed successfully with no affect on the integrity of the backup file. This invalid warning will no longer be output by RMU/BACKUP.

The following example shows that an invalid %RMU−W−BADPTLARE warning message could be output by Oracle Rdb RMU/BACKUP for database pages that were deleted by a SQL TRUNCATE TABLE statement.

```
$ rmu/backup/nolog test.rdb testfull.rbf
$ sql
 attach 'filename test.rdb';
 truncate table t;
 commit;
 insert into t values (1);
1 row inserted
 commit;
 exit
$ rmu/backup/incremental/nolog test.rdb testinc.rbf
$ sql
 drop database filename test;
 exit
$ rmu/restore/nocdd/nolog testfull.rbf
%RMU−I−AIJRSTAVL, 0 after−image journals available for use
%RMU−I−AIJISOFF, after−image journaling has been disabled
%RMU−W−USERECCOM, Use the RMU Recover command. The journals are not available.
$ rmu/restore/incremental/nocdd/noconfirm/nolog testinc.rbf
%RMU−W−USERECCOM, Use the RMU Recover command. The journals are not available.
$ rmu/backup/nolog test.rdb testfull.rbf
%RMU−W−BADPTLARE, invalid larea for uniform data page 6 in storage area 2
%RMU−W−BADPTLAR2,                   SPAM larea_dbid: 0, page larea_dbid: 47
```

This problem has been corrected in Oracle Rdb Release 7.2.0.2.

3.4 Row Cache Errors Fixed

3.4.1 Row Cache Latching Enhancements and Corrections

In prior releases of Oracle Rdb, it was possible for Row Cache hash latches to be incorrectly held causing "hangs". To help avoid these problems, the two latching mechanisms within the Row Cache feature have been corrected to help eliminate possible race conditions and errant latches without matching unlatches.

In addition, for those hash latches that experience higher levels of contention, the built−in stall timer used between polls of the latch has been reduced to allow more responsive detection of the latch being released.

Customers are reminded that setting caches to "ROW REPLACEMENT IS DISABLED" allows multiple processes to scan internal row cache hash chains simultaneously. This can improve cache search performance for heavily utilized caches.

Finally, a new SHOW STATISTICS screen "Cache Latch Information" may provide additional debugging information.

These problems have been corrected in Oracle Rdb Release 7.2.0.2.

3.4.2 Invalid Log File Logical Name Causes RCS to Terminate

Bug 5125792

In prior versions of Oracle Rdb, the Row Cache Server (RCS) process could fail to correctly start if the RCS log file was unable to be created.

For example, if the "RDM$BIND_RCS_LOG_FILE" logical name was defined with an invalid or inaccessible device or directory specification, the RCS process could fail while starting. The monitor log file would contain an entry "%RDMS−F−RCSABORTED, record cache server process terminated abnormally " and user processes would be terminated with the status "%RDMS−F−TERMINATE, database recovery failed−−−access to database denied by monitor".

This problem has been corrected in Oracle Rdb Release 7.2.0.2. The Row Cache Server (RCS) process now matches the behavior of the other database server processes (such as the database recovery service (DBR)) and will continue running without a log file if the log file is not able to be created.

3.5 RMU Show Statistics Errors Fixed

3.5.1 Incorrect Journal "CurrEof" Displayed by RMU/SHOW STATISTICS

Bug 5195930

In Oracle Rdb Release 7.2.0.1, the RMU/SHOW STATISTICS utility would show the physical size of the journal instead of the actual current journal end−of−file. For example, the following shows the EOF at 5120 when it should have shown a smaller number:

```
Node: RANDM4 (1/1/1)   Oracle Rdb V7.2−011 Perf. Monitor 29−APR−2006 18:00:42.47
Rate: 3.00 Seconds          AIJ Journal Information         Elapsed: 00:00:39.78
Page: 1 of 2  $1$DGA58:[RDB_RANDOM.RDB_RANDOM_SU_12_C2]RNDDB.RDB;1  Mode: Online
−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
Journaling: enabled   Shutdown:   60  Notify: disabled  State: Accessible
ALS: Manual     ABS: disabled  ACE:           FC: enabled   CTJ: enabled
ARB.Count:  300 ARB.Avail:  300 SwtchSched:  0 NxtSwtch:
After−Image.Journal.Name....... SeqNum   AIJsize   CurrEOF Status. State.......
J1                                 151      5120      5120 Current Accessible
```

The RMU/DUMP/HEADER=JOURNAL command can be used to get the correct journal end−of−file.

This problem has been corrected in Oracle Rdb Release 7.2.0.2.

3.5.2 Active User Count Incorrect as ABS Starts and Stops

Bug 5134756

In prior versions of Oracle Rdb, the statistics counter "NUM_ACTIVE" was not correctly decremented when the AIJ Backup Server (ABS) process completed a backup. This would lead to an ever−increasing value for the number of users as shown by the RMU /SHOW STATISTICS utility.

This problem has been corrected in Oracle Rdb Release 7.2.0.2. The AIJ Backup Server (ABS) process correctly adjusts the active user counter when it exits.

Chapter 4
Software Errors Fixed in Oracle Rdb Release 7.2.0.1

This chapter describes software errors that are fixed by Oracle Rdb Release 7.2.0.1.

4.1 Software Errors Fixed That Apply to All Interfaces

4.1.1 Poor Performance Using Database With Many Logical Areas

Bug 4917868

If a database ever had more than 512 logical areas defined, and the maximum number of logicals areas was an odd number, queries could consume considerably more CPU when executing.

For example, in the output from the RMU/DUMP/HEADER command look for the "Logical area count":

```
$ RMU/DUMP/HEADER SLOW_DATABASE.RDB
...
    Logical area count is 6073
...
```

In the above example, the logical area count is an odd number. Executing a complex query against that database results in the following perfomance statistics:

```
ELAPSED:  0 00:01:53.10  CPU: 0:01:52.83  BUFIO: 12  DIRIO: 107500 FAULTS: 4100
```

After adding enough tables, the number of logical areas becomes even:

```
SQL> ATTACH 'FILENAME SLOW_DATABASE';
SQL> CREATE TABLE TEMP (C1 INT);
SQL> COMMIT;
SQL> DROP TABLE TEMP;
SQL> COMMIT;
$ RMU/DUMP/HEADER SLOW_DATABASE.RDB
...
    Logical area count is 6074
...
```

The same query now shows the following performance statistics:

```
ELAPSED: 0 00:00:21.18  CPU: 0:00:20.28  BUFIO: 12  DIRIO: 107473  FAULTS: 4205
```

This problem was due to alignment faults that were occurring when incrementing database statistics counters. The counters were not properly aligned if the maximum number of logical areas was an odd number.

This problem can be prevented by forcing the number of logical areas to be an even number as shown above.

This problem has been corrected in Oracle Rdb Release 7.2.0.1.

4.1.2 Monitor Bugchecks at MON$SEND_REPLY + 0000008C

Bug 4961487

If a database used the AIJ Log Server (ALS) or Row Cache Server (RCS) processes, and the database monitor encountered an error when attempting to start those servers, then the monitor could fail with a bugcheck similar to the following:

```
***** Exception at 000E0BBC : MON$SEND_REPLY + 0000007C
%COSI-F-BUGCHECK, internal consistency failure
```

Examination of the monitor logfile would show that the monitor could not start a server. For example:

```
- sending user attach reply to 0000D369:1
  - "%RDMS-F-CANTCREALS, error creating AIJ Log Server process"
  - "-SYSTEM-F-NOSLOT, no PCB available"
```

After encountering the error, the next attempt to attach to the database would result in the bugcheck.

This problem occurred because the monitor neglected to delete the data structure representing the user that had the failed attach attempt. When the server startup failed, an error processing path was used that did not properly delete the structure. When the monitor again attempted to send messages to waiting users, it would attempt to send to the same user again but that user was not in a state that would allow another message, resulting in the monitor bugcheck.

This is an exceptionally rare problem and would typically only be encountered when the system was low on resources.

This problem has been corrected in Oracle Rdb Release 7.2.0.1.

4.1.3 Memory Leak in Attach/Detach

Bug 4866466

Every time a process would attach and detach from a database without exiting the main image, at least 104 bytes of memory would be lost.

For example, if the following commands were repeatedly executed, memory usage would slowly increase:

```
SQL> ATTACH 'FILENAME PERSONNEL';
SQL> SHOW TABLE
SQL> DISCONNECT ALL;
SQL> ATTACH 'FILENAME PERSONNEL';
SQL> SHOW TABLE
SQL> DISCONNECT ALL;
...
```

The only way to avoid the problem is to periodically rundown the main image and restart the application.

This problem has been corrected in Oracle Rdb Release 7.2.0.1.

4.1.4 Aggregate Query with ORDER BY Clause Bugchecks

Bugs 4747999, 2649215, 3194445 and 3357593

The aggregate query bugchecks when two derived tables (or views) are joined via two indexed columns, followed by an ORDER BY clause on one of the indexed columns. The following example shows the bugcheck.

```
SELECT count(*)
    (select * from test_1) as V1,
    (select * from test_2) as V2
  WHERE
    V2.A_DATE     =  V1.A_DATE AND
    V2.MEMBER     =  V1.MEMBER AND
    V1.CURRENCY   = 'EUR'
    ORDER BY V1.TSN_TYPE;
%RDMS-I-BUGCHKDMP, generating bugcheck dump file DIR:RDSBUGCHK.DMP;
%RDB-F-BUG_CHECK, internal consistency check failed
```

As a workaround, the query works if the aggregate count function is replaced by the simple select on the columns, as in the following example.

```
SELECT *
    (select * from test_1) as V1,
    (select * from test_2) as V2
  WHERE
    V2.A_DATE     =  V1.A_DATE AND
    V2.MEMBER     =  V1.MEMBER AND
    V1.CURRENCY   = 'EUR'
    ORDER BY V1.TSN_TYPE;
Tables:
  0 = TEST_1
  1 = TEST_2
Sort: 0.TSN_TYPE(a)
Cross block of 2 entries
  Cross block entry 1
    Merge of 1 entries
      Merge block entry 1
      Leaf#01 BgrOnly 0:TEST_1 Card=10
        Bool: 0.CURRENCY = 'EUR'
        BgrNdx1 INDEX_TEST_1 [0:0] Fan=9
          Bool: 0.CURRENCY = 'EUR'
  Cross block entry 2
    Merge of 1 entries
      Merge block entry 1
      Conjunct: (1.A_DATE = 0.A_DATE) AND (1.MEMBER = 0.MEMBER)
      Get     Retrieval by index of relation 1:TEST_2
        Index name  INDEX_1_TEST_2 [2:2]       Direct lookup
          Keys: (1.A_DATE = 0.A_DATE) AND (1.MEMBER = 0.MEMBER)
0 rows selected
```

This problem has been corrected in Oracle Rdb Release 7.2.0.1.

4.1.5 Wrong Result from Query with Zigzag Match and Reverse Scan

Bug 4771936

The following query with zigzag match strategy using reverse scan returns the wrong result (should return one row).

```
SET FLAGS 'STRATEGY,DETAIL';
SELECT T2.ORDER_NO,T1.CUST_NO
FROM TT2 T2, TT1 T1
WHERE T1.ORDER_NO =  T2.ORDER_NO
  AND T1.CUST_NO = 123
;
~S: Outline "BUG_DESC_OUTLINE" used
Tables:
  0 = TT2
  1 = TT1
Conjunct: 1.ORDER_NO = 0.ORDER_NO
Match
  Outer loop      (zig-zag)
    Index only retrieval of relation 1:TT1
      Index name  TT1_NDX_DESC [1:1]  Reverse Scan
        Keys: 1.CUST_NO = 123
  Inner loop      (zig-zag)
    Index only retrieval of relation 0:TT2
      Index name  TT2_NDX [0:0]
0 rows selected

where the tables contain the following:

select * from tt1;
Tables:
  0 = TT1
Get     Retrieval by index of relation 0:TT1
  Index name  TT1_NDX_DESC [0:0]
 ORDER_NO         CUST_NO   COMPLETED_DATE
 0000047          123   31-DEC-9999 00:00:00.00
 0000046          123   13-OCT-2005 17:00:34.03
 0000044          123   31-DEC-9999 00:00:00.00
3 rows selected

select * from tt2;
Tables:
  0 = TT2
Index only retrieval of relation 0:TT2
  Index name  TT2_NDX [0:0]
 ORDER_NO
 0000045
 0000046
2 rows selected
```

As a workaround, the query works if the reverse scan is disabled, as in the following example.

```
SQL> set flags 'noreverse_scan'
SQL> .... execute the same above query here ...
Tables:
  0 = TT2
  1 = TT1
Conjunct: 1.ORDER_NO = 0.ORDER_NO
Match
  Outer loop
    Sort: 1.ORDER_NO(a)
    Index only retrieval of relation 1:TT1
      Index name  TT1_NDX_DESC [1:1]
        Keys: 1.CUST_NO = 123
  Inner loop      (zig-zag)
    Index only retrieval of relation 0:TT2
      Index name  TT2_NDX [0:0]
 T2.ORDER_NO    T1.CUST_NO
```

```
 0000046        123
1 row selected
```

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

1. The main select query joins two tables using a match strategy with zigzag skip on both inner and outer legs.
2. The join key is a descending index segment in the outer index but an ascending segment in the inner index.
3. The reverse scan is applied on the index retrieval at the outer leg.
4. The leading segment of the outer index is used as a filter predicate with an equality clause.

This problem has been corrected in Oracle Rdb Release 7.2.0.1.

4.1.6 GROUP BY Query Bugchecks in Zigzag Strategy

Bug 4459069

The following query with GROUP BY clause bugchecks in zigzag strategy.

```
select  s.SALARY_END, x.B
      from XXX x,
          SALARY_HISTORY s
      where
            x.A =  s.SALARY_AMOUNT and
            s.EMPLOYEE_ID <= '00164'
            AND  s.SALARY_AMOUNT =  0
      group by s.SALARY_END, x.B  ;
%RDMS-I-BUGCHKDMP, generating bugcheck dump file DIR:RDSBUGCHK.DMP;
%RDB-F-BUG_CHECK, internal consistency check failed
```

As a workaround, the query works if the sql flag 'zigzag_outer' is disabled, as in the following example.

```
set flags 'nozigzag_outer';
select  s.SALARY_END, x.B
      from XXX x,
          SALARY_HISTORY s
      where
            x.A =  s.SALARY_AMOUNT and
            s.EMPLOYEE_ID <= '00164'
            AND  s.SALARY_AMOUNT =  0
      group by s.SALARY_END, x.B  ;
Tables:
  0 = XXX
  1 = SALARY_HISTORY
Reduce: 1.SALARY_END, 0.B
Sort: 1.SALARY_END(a), 0.B(a)
Conjunct: 0.A = 1.SALARY_AMOUNT
Match
  Outer loop
    Index only retrieval of relation 0:XXX
      Index name  XXX_I [1:1]
        Keys: 0.A = 0
  Inner loop      (zig-zag)
    Conjunct: 1.EMPLOYEE_ID <= '00164'
    Conjunct: 1.SALARY_AMOUNT = 0
    Get     Retrieval by index of relation 1:SALARY_HISTORY
```

```
      Index name  SH_EMPLOYEE_ID [0:1]
        Keys: 1.EMPLOYEE_ID <= '00164'
0 rows selected
```

This problem has been corrected in Oracle Rdb Release 7.2.0.1.

4.1.7 Wrong Result from Left Outer Join with OR Predicates

Bug 5038264

A query with multiple UNION and LEFT OUTER JOINs containing a CASE expression with OR predicates, returns the wrong result in Rdb Release 7.1.4.3. The query should return the value 'Y' for the column NOTES_IND, but it returns the value 'F'.

After simplifying the query, the problem is still reproduced without any UNION clauses and the multiple LEFT OUTER JOINs are reduced down to a single join. The following is the simplified version of the original query from the customer.

```
select
    case
        when (exists
            (select *
            from notes n
            where
                (n.item_id = tt.media_hc_id and
                 n.item_type_ind = tt.media_hc_auto_ind )
                    or
                (n.item_id = tt.cpty_hc_id and
                 n.item_type_ind = tt.cpty_hc_auto_ind )
                 ))
        then 'Y' else 'N'
    end notes_ind,
    tt.trans_unit_id,
    tt.cpty_hc_auto_ind,
    tt.media_hc_auto_ind
from (
    select
        d.trans_unit_id,
        d.cpty_hc_auto_ind,
        d.cpty_hc_id,
        d.media_hc_auto_ind,
        d.media_hc_id,
        d.media_hc_ver_id
    from
        process_trans p, dealer d
    where
            d.company_id = 'BTNYC'
        and p.trans_unit_id = d.trans_unit_id
        and p.group_id = 2
    ) as tt
    left outer join confo c
    on
       (c.confo_in_man_id = tt.media_hc_id
        and c.ver_id = tt.media_hc_ver_id
        and tt.media_hc_auto_ind = 'M')
    where
         (c.trade_date >= '00000000');
 NOTES_IND   TT.TRANS_UNIT_ID   TT.CPTY_HC_AUTO_IND   TT.MEDIA_HC_AUTO_IND
```

```
 N                  400000022   A                     M
```

If we turn on the sql flags 'strategy, detail', we can see the obvious problem in the strategy output.

```
Tables:
  0 = PROCESSING_GROUP_TRANS_UNIT
  1 = DEAL_FOLDER
  2 = CONFO_IN_MAN
  3 = NOTES
Cross block of 2 entries
  Cross block entry 1
    Conjunct: 2.TRADE_DATE >= '00000000'
    Conjunct: 2.TRADE_DATE >= '00000000'
    Match    (Left Outer Join)
      Outer loop
        Sort: 1.MEDIA_HC_ID(a), 1.MEDIA_HC_VER_ID(a)
        Merge of 1 entries
          Merge block entry 1
          Cross block of 2 entries
            Cross block entry 1
              Leaf#01 BgrOnly 1:DEAL_FOLDER Card=3103449
                Bool: 1.COMPANY_ID = 'BTNYC'
                BgrNdx1 DEAL_FOLDER_MATCH_IDX [1:1] Fan=6
                  Keys: 1.COMPANY_ID = 'BTNYC'
            Cross block entry 2
              Index only retrieval of relation 0:PROCESSING_GROUP_TRANS_UNIT
                Index name  PROC_GROUP_TRANS_UNIT_IDX [2:2]    Direct lookup
                  Keys: (0.PROCESSING_GROUP_ID = 2) AND (0.TRANS_UNIT_ID =
                        1.TRANS_UNIT_ID)
      Inner loop      (zig-zag)
        Conjunct: 2.TRADE_DATE >= '00000000'
        Get     Retrieval by index of relation 2:CONFO_IN_MAN
          Index name  CONFO_IN_MAN_IDX [0:0]
  Cross block entry 2
    Aggregate-F1: 0:COUNT-ANY (<subselect>)
    Conjunct: ((3.ITEM_ID = 1.MEDIA_HC_ID) AND (3.ITEM_TYPE_IND =
              1.MEDIA_HC_AUTO_IND)) OR ((3.ITEM_ID = 1.CPTY_HC_ID) AND (
              3.ITEM_TYPE_IND = 1.CPTY_HC_AUTO_IND))
    OR index retrieval                                  <== See Note
      Index only retrieval of relation 3:NOTES
        Index name  NOTES_IDX [2:2]
          Keys: (3.ITEM_TYPE_IND = 1.MEDIA_HC_AUTO_IND) AND (3.ITEM_ID =
                1.MEDIA_HC_ID)
 NOTES_IND   TT.TRANS_UNIT_ID   TT.CPTY_HC_AUTO_IND   TT.MEDIA_HC_AUTO_IND
 N                  400000022   A                     M
1 row selected
```

Note that the static OR index retrieval contains only the first leg. The second leg of the static OR retrieval is missing in the above strategy and thus causes the query to return the wrong result.

There is no known workaround for this problem.

This problem has been corrected in Oracle Rdb Release 7.2.0.1.

4.1.8 ACCVIO from RDMS$$CREATE_EGET Generated Code

Bug 4913301

Certain complex queries could cause an ACCVIO similar to the following.

```
***** Exception at 01850960 : symbol not found
%SYSTEM-F-ACCVIO, access violation, reason mask=00, virtual
                  address=0000000020578208, PC=0000000001850960, PS=00000009
Saved PC = 00F050CC : RDMS$$EXE_ACTION + 000027CC
Saved PC = 0184FD2C : symbol not found
Saved PC = 00F236B8 : RDMS$TOP_START_REQUEST + 00000898
Saved PC = 00D00444 : BLI$CALLG + 000000BC
Saved PC = 0114D1E4 : KOD$SETSTK_AND_CONTINUE + 0000019C
```

This problem is more likely to occur when using SQL*Net for Rdb or after issuing a "SET DIALECT 'ORACLE LEVEL2';" statement.

A possible workaround might involve not using SQL*Net for Rdb or not using "SET DIALECT 'ORACLE LEVEL2';".

This problem has been corrected in Oracle Rdb Release 7.2.0.1.

4.1.9 Bugchecks or Corruption of Ranked Indexes

Bug 5040585

It was possible during insert operations on tables with ranked indexes that a bugcheck could occur in *PSIINDEX2JOINSCR*. It is suspected that the same problem could introduce index corruptions if the internal consistency check did not detect the problem.

The problem is more likely to happen if the index is very deep and the number of buffers being used is very small or the index is in a row cache.

The following is an example of the bugcheck footprint when the error occurs.

```
COSI-F-BUGCHECK, internal consistency failure
Exception occurred at PSIINDEX2JOINSCR + 000002D8
Called from PSII2BALANCE + 000015F8
Called from PSII2INSERTT + 00000838
Called from PSII2INSERTT + 0000064C
Bugcheck when working on sorted ranked index
Running image SQLU711.EXE
Dump created: 27-FEB-2006 22:15:55.32
GETLKI section omitted.
Database root: $1$DGA147:[MONEIL.5068361993]ISRS_DB
This bugcheck may have been caused by a corrupt index.
The database should be verified to check for such corruption.
Suggested command: RMU/VERIFY/INDEX $1$DGA147:[MONEIL.5068361993]ISRS_DB
```

There is no known workaround for this problem.

The problem is extremely rare and depends heavily on the index internal structure. For this reason, if the index was dropped and recreated it would most likely eliminate the error.

This problem has been corrected in Oracle Rdb Release 7.2.0.1.

4.2 SQL Errors Fixed

4.2.1 Unexpected UNRES_AREAX Error After Using TRUNCATE TABLE

Bug 4998128

The following example shows that TRUNCATE TABLE and SET TRANSACTION ... RESERVING PARTITION do not always function correctly when used in the same session.

```
SQL> truncate table TEST_TABLE;
SQL> commit;
SQL>
SQL> insert into TEST_TABLE ( A_CODE) values ( '~~~~~' );
1 row inserted
SQL> commit;
SQL>
SQL> set transaction read write
cont>     reserving TEST_TABLE partition( 2 ) for EXCLUSIVE WRITE;
SQL>
SQL> select * from TEST_TABLE
cont>   where A_CODE >  'DTE  ' and A_CODE <= 'HEN3 ';
%RDB-E-UNRES_REL, relation TEST_TABLE in specified request is not a relation
reserved in specified transaction
-RDMS-E-UNRES_AREAX, area 74 within index "TEST_TABLE_IDX2" is not reserved
```

The problem here is that the index scan code tries to read a non−NULL root node for the index TEST_TABLE_IDX2 so that the range query can be executed to locate the source partitions (that contain requested data). However, TRUNCATE TABLE does not leave any root nodes: these are added when the first INSERT is executed and the final partition is selected instead. In this case, the partition was not included in the RESERVING clause and the query failed.

This problem has been corrected in Oracle Rdb Release 7.2.0.1. The TRUNCATE TABLE statement now writes back an empty root node for each SORTED or SORTED RANKED index partition.

4.2.2 Unnecessary FOREIGN KEY Constraints Evaluated by TRUNCATE TABLE

Bug 2893577

In prior versions of Rdb, the TRUNCATE TABLE statement would validate constraints after the data was erased from the table. While specific constraints such as NOT NULL, UNIQUE and PRIMARY were excluded, the FOREIGN KEY constraints were still processed. As these often referenced other tables, I/O was expended unnecessarily.

This simple example shows that the FOREIGN KEY (REFERENCES) constraint F_I1 is evaluated even when the removal of rows does not affect other tables.

```
SQL> create table t1(i1 int constraint t1_p primary key not deferrable);
SQL> create table t2(i1 int constraint f_i1 references t1(i1) not deferrable);
```

```
SQL> insert into t1 values (1);
1 row inserted
SQL> insert into t2 values (1);
1 row inserted
SQL> set flags 'item_list,strategy,internal';
SQL> truncate table t2;
~H Extension (VERIFY CONSTRAINTS) Item List: (len=9)
0000 (00000) RDB$K_EXT_VFYC_EXCLUDE_UNIQUE
0003 (00003) RDB$K_EXT_VFYC_TABLE_NAME "T2"
0008 (00008) RDB$K_INFO_END
~H: ...verify constraint "F_I1"
Cross block of 2 entries
  Cross block entry 1
    Conjunct        Get     Retrieval sequentially of relation T2
  Cross block entry 2
    Conjunct        Aggregate-F1    Conjunct        Get
    Retrieval sequentially of relation T1
SQL>
```

This problem has been corrected in Oracle Rdb Release 7.2.0.1. TRUNCATE TABLE no longer evaluates FOREIGN KEY constraints defined for the table being truncated. However, FOREIGN KEY constraints for other tables that reference this table will still be evaluated.

4.2.3 TRACE Statement Now Trims Trailing ASCII Space Characters

Bug 2814186

This release changes the output of the TRACE statement so that trailing ASCII space characters are trimmed from the result. The trace output buffer is considered to be a CHAR(512) string with character set UNSPECIFIED. If the string contains data derived from other character sets, such as those that use a different space character, then those spaces will not be trimmed by TRACE.

This problem has been corrected in Oracle Rdb Release 7.2.0.1.

4.2.4 SHOW TABLE Now Supports Synonyms for Views

Bug 4454982

The SHOW TABLE command will display details of both tables and views. When SHOW TABLE is presented with a synonym for a view, the command fails to locate the view. This problem has been corrected in Oracle Rdb Release 7.2.0.1. SHOW now displays the view details if given the name of a synonym for a view.

4.2.5 SQLBUGCHK With 'SQL$SEMMSC − 3' After DB Creation Failure

If a CREATE DATABASE statement with the MULTISCHEMA IS ON option failed, subsequent attempts to create a database in the same session would receive an error message similar to the following:

```
%RDMS-I-BUGCHKDMP, generating bugcheck dump file MYDISK:[MYDIR]SQLBUGCHK.DMP;
%SQL-F-BUGCHK, There has been a fatal error. Please contact your Oracle support
```

```
representative. SQL$SEMMSC − 3
```

The resulting SQLBUGCHK.DMP would have a signature similar to the following:

```
***** Exception at 0017B824 : SQL$$INSERT_SYMBOL + 00000034
%SQL−F−BUGCHK, There has been a fatal error. Please contact your Oracle support
representative. SQL$SEMMSC − 3
```

The following example shows a CREATE DATABASE statement which fails because the remote node is unreachable. The second CREATE DATABASE statement should succeed but instead fails because of the problem described in this note.

```
$ SQL$
SQL> −− create a database on a node that doesn't exist.
SQL> create database filename dfabjd::foo multischema is on;
%SQL−F−ERRCRESCH, Error creating database filename dfabjd::foo
−RDB−F−IO_ERROR, input or output error
−SYSTEM−F−NOSUCHNODE, remote node is unknown
−− Now this one should work.
SQL> create database filename bdmscr7 multischema is on;
%RDMS−I−BUGCHKDMP, generating bugcheck dump file MYDISK:[MYDIR]SQLBUGCHK.DMP;
%SQL−F−BUGCHK, There has been a fatal error. Please contact your Oracle support
representative. SQL$SEMMSC − 3
```

As a workaround to this problem, restart SQL and re−enter the CREATE DATABASE statement.

This problem has been corrected in Oracle Rdb Release 7.2.0.1.

4.2.6 SQL$MOD SQLBUGCHK With SQL$$QUEUE_SEND + 00000284

Bug 4594637

If a SQL Module Language module defines a procedure which requires a default database and there is no default database, SQL$MOD might fail and produce the following message:

```
%RDMS−I−BUGCHKDMP, generating bugcheck dump file MYDISK:[MYDIR]SQLBUGCHK.DMP;
```

The resulting SQLBUGCHK.DMP would have a signature similar to the following:

```
***** Exception at 0041B4C4 : SQL$$QUEUE_SEND + 00000284
%SYSTEM−F−ACCVIO, access violation, reason mask=00, virtual address=
00000000000000CB, PC=000000000041B4C4, PS=0000001B
```

The following example shows a SQL Module Language program which has a procedure (CORE_GET_NUMBER_OF_DAYS) which requires a default database. This is because the variables CURRENT_CASH_DATE and ROWS_READ rely on a default database for their types to be resolved.

```
module test_module
  language general
   parameter colons
declare scp_db_handle alias for filename 'foo'
declare transaction read only
procedure core_get_x_number_of_days
```

```
     sqlca,
     :x_number_of_days     char(2),
     :date_param           date vms;
begin
      declare :current_cash_date      date_domain;
      declare :rows_read              count_domain;

end;
```

SQL$MOD will now report %SQL−F−NODEFDB for each of the variables which need a default database for type resolution and then terminate normally.

As a workaround for this problem, correct the type definition of the variables.

This problem has been corrected in Oracle Rdb Release 7.2.0.1.

4.2.7 Unexpected DEADLOCK Returned by CREATE TABLE

Bug 5060366

In prior versions of Oracle Rdb, CREATE TABLE might report an unexpected DEADLOCK error if the CREATE TABLE statement included constraint definitions.

The following example shows that the second CREATE TABLE fails with a deadlock.

```
SQL> create table X0 (a integer not null not deferrable);
SQL> create table X1 (a integer not null not deferrable);
%RDB−E−DEADLOCK, request failed due to resource deadlock
−RDB−E−NO_META_UPDATE, metadata update failed
−RDMS−F−DEADLOCK, deadlock on client '........X1  '
202031580000001E0000000400000055
```

This problem only occurs when the allocated RDB$RELATION_ID exceeds the maximum allowed (8182). In this case, Rdb attempts to reuse an old value previously assigned to a table and freed by a DROP TABLE.

This problem has been corrected in Oracle Rdb Release 7.2.0.1. Rdb now correctly locates an unused relation id value without the deadlock error being reported.

4.3 RMU Errors Fixed

4.3.1 RMU/RECOVER Bugchecks in KUTREC$ABORT

When recovering journals, it was possible for the RMU/RECOVER command to fail with an error similar to the following:

```
%RMU-E-RECFAILED, fatal, unexpected roll-forward error detected at AIJ record
1796212
%COSI-F-BUGCHECK, internal consistency failure
%RMU-F-FATALOSI, Fatal error from the Operating System Interface.
%RMU-F-FTL_RCV, Fatal error for RECOVER operation at  8-DEC-2005 08:04:39.41
```

The bugcheck dump contained the following exception:

```
***** Exception at 0071D198 : KUTREC$ABORT + 000005D8
%COSI-F-BUGCHECK, internal consistency failure
```

Examination of the bugcheck dump showed that there were journal entries that could not be applied:

```
$ SEARCH RMUBUGCHK.DMP "FAIJBL @"
  FAIJBL @00C8BD40:         MSN = 0.                  PSN = 0.
  FAIJBL @00C8B900:         MSN = 0.                  PSN = 0.
```

This particular problem would only occur when the fast commit *CHECKPOINT TIMED EVERY n SECONDS* feature was being used and the following events occurred:

1. A transaction made updates to the database.
2. After the last change was made, but before the transaction was committed, the checkpoint timer expired causing the checkpoint location to be set to "none".
3. The process began committing the transaction, but was abnormally terminated after writing a commit entry to the after-image journal and before updating its TSNBLK entry in the database root (.RDB) file.

When the above sequence of events occurred, the database recovery process (DBR) would examine the last checkpoint location and mistakenly determine that the process had not committed the transaction since there was no checkpoint for the failed user. Consequently, the DBR would rollback the transaction. However, the journal would still show that the transaction had committed even though the changes were no longer in the database. If the database was later restored, and the journal was applied using RMU/RECOVER, RMU would attempt to apply the changes that were rolled back by the DBR to the database. However, it might not have been possible to apply those changes since the space freed when the DBR rolled back the transaction got reused by other transactions. That would cause the RMU/RECOVER command to fail.

This problem can be avoided by disabling the *CHECKPOINT TIMED EVERY n SECONDS* feature.

This problem has been corrected in Oracle Rdb Release 7.2.0.1.

4.3.2 RMU REPAIR Did Not Retry Root File Open for Offline Access

Bug 4899677

A problem has been fixed which caused the RMU/REPAIR call to open the Oracle Rdb database root file for offline access to not retry the open if the open failed because the database was in use. Now the standard open retry behavior used elsewhere in RMU will be used of retrying the offline open of the database root file for approximately two minutes before giving up and returning an error.

The following example shows the problem where the retry of the open of the database root file for offline access was not repeated for approximately two minutes before returning a fatal error.

```
$ rmu/repair/spams mf_personnel.rdb
 %RMU-I-WAITOFF, Waiting for offline access to MF_PERSONNEL.RDB
 %RMU-F-FILACCERR, error opening database root file MF_PERSONNEL.RDB
 -COSI-E-FLK, file currently locked by another user
 %RMU-F-FTL_REP, Fatal error for REPAIR operation at 18-JAN-2006 13:11:52.89
```

The following example shows the corrected behavior where the retry of the open of the database root file for offline access is repeated for approximately two minutes before an error is returned.

```
$ rmu/repair/spams mf_personnel.rdb
%RMU-I-WAITOFF, Waiting for offline access to MF_PERSONNEL.RDB
%RMU-I-WAITOFF, Waiting for offline access to MF_PERSONNEL.RDB
%RMU-I-WAITOFF, Waiting for offline access to MF_PERSONNEL.RDB
%RMU-I-WAITOFF, Waiting for offline access to MF_PERSONNEL.RDB
%RMU-I-WAITOFF, Waiting for offline access to MF_PERSONNEL.RDB
%RMU-I-WAITOFF, Waiting for offline access to MF_PERSONNEL.RDB
%RMU-F-FILACCERR, error opening database root file MF_PERSONNEL.RDB
-COSI-E-FLK, file currently locked by another user
%RMU-F-FTL_REP, Fatal error for REPAIR operation at 18-JAN-2006 13:11:52.89
```

This problem has been corrected in Oracle Rdb Release 7.2.0.1.

4.3.3 Incorrect Cardinalities or Process Termination from RMU Load

Bug 3528247

When using the RMU Load utility with the *DEFER_INDEX_UPDATES* qualifier, it was possible that the process would terminate with an access violation status. Even if the load completed sucessfully, the index and index prefix cardinalities could be highly incorrect.

Index updates can be deferred for sorted indexes that are not unique and also not used for placement via in the corresponding storage map.

In the following example, four rows are loaded. However there are only two unique key values. In fact, both the index cardinality and the prefix cardinality should be two.

```
SQL> create index i1 on t1 (f1, f2);
SQL> commit;
```

```
SQL>  Exit
$ reca rmu/load
$ rmu/load/defer_index testdb t1 t1
%RMU-I-DATRECREAD,  4 data records read from input file.
%RMU-I-DATRECSTO,   4 data records stored.
$ rmu/show optimizer_statistics/table=t1 testdb
    ...
Optimizer Statistics for table : T1

  Cardinality            : 4
  Row clustering factor  : 0.0000000

Index name : I1
  Index Cardinality      : 4
  Average Depth          : 0.0000000
  Key clustering factor  : 0.0000000
  Data clustering factor : 0.0000000
  Segment Column                 Prefix cardinality
    F1                              4
    F2                              0
$ SQL
SQL> select * from t1;
          F1            F2
           1             1
           1             1
           2             2
           2             2
4 rows selected
```

While this problem will not cause wrong results or data corruption, it may lead the optimizer to choose a less optimal retrieval strategy in some cases.

If the process terminates during the load operation, the load will be incomplete. In this case, the load must be restarted without the DEFER_INDEX_UPDATES qualifier. If the load completes, the index and prefix cardinalities will be incorrect for all deferred indices. This can be corrected using the *RMU/COLLECT OPTIMIZER_STATISTICS* command as shown in the following example.

```
$ rmu/collect optimizer_statistics/statistic=cardinality/index=i1 testdb
Start loading tables... at 18-JAN-2006 23:46:14.23
Done loading tables.... at 18-JAN-2006 23:46:14.52
Start loading indexes... at 18-JAN-2006 23:46:14.52
Done loading indexes.... at 18-JAN-2006 23:46:14.66
Start collecting btree index stats... at 18-JAN-2006 23:46:15.78
Done collecting btree index stats.... at 18-JAN-2006 23:46:15.79
Start collecting table & hash index stats... at 18-JAN-2006 23:46:15.79
Done collecting table & hash index stats.... at 18-JAN-2006 23:46:15.80
Start calculating stats... at 18-JAN-2006 23:46:15.91
Done calculating stats.... at 18-JAN-2006 23:46:15.91
Start writing stats... at 18-JAN-2006 23:46:16.65
    ...
Optimizer Statistics collected for table : T1

  Cardinality            : 4

Index name : I1
  Index Cardinality      : 2
  Segment Column                 Prefix cardinality
    F1                              2
    F2                              0
Done writing stats.... at 18-JAN-2006 23:46:16.85
```

This problem has been corrected in Oracle Rdb Release 7.2.0.1.

4.3.4 RMU/OPEN Maximum Global Buffer Count Check Corrected

In the prior Oracle Rdb V7.2 release, the RMU/OPEN command incorrectly limited the maximum allowed global buffer count to 524,288 rather than the expected value of 1,048,576.

This problem has been corrected in Oracle Rdb Release 7.2.0.1.

4.3.5 RMU/MOVE_AREA Enabled AIJ When it had Been Disabled

Bug 4861904

A problem has been fixed which caused the Oracle Rdb RMU/MOVE_AREA command to enable AFTER IMAGE JOURNALING in the moved database when it was disabled in the database before the move, and no AIJ related qualifiers (AFTER_JOURNAL, AIJ_OPTIONS) had been specified in the RMU/MOVE_AREA command which would alter the state of AFTER IMAGE JOURNALING for the moved database. This was caused by the AIJ Enabled state of the database always being set to ENABLED after the move. This has been corrected. Now the AIJ Enabled state of the original database will be preserved unless the AFTER_JOURNAL or AIJ_OPTIONS qualifiers have been used to change the AIJ Enabled state of the database.

The following example shows the problem where the RMU/MOVE_AREA command reenabled AFTER IMAGE JOURNALING in the moved database even though it had originally been disabled before the move.

```
$rmu/set after_journal/add=(name=aij1, file=aij1) mf_personnel
%RMU-W-AIJDEVDIR, AIJ filename "AIJ1" does not include a device/directory
%RMU-I-LOGCREAIJ, created after-image journal file DEVICE:[DIRECTORY]AIJ1.AIJ;1
%RMU-I-LOGMODSTR,     added after-image journal definition "AIJ1"
%RMU-W-DOENBLAIJ, after-image journaling must be enabled to ensure recovery
$rmu/set after_journal/enable mf_personnel
%RMU-I-LOGMODSTR,     activated after-image journal "AIJ1"
%RMU-I-LOGMODFLG,     enabled after-image journaling
%RMU-W-DOFULLBCK, full database backup should be done to ensure future recovery
$rmu/set after_journal/disable mf_personnel
%RMU-I-LOGMODFLG,     disabled after-image journaling
$rmu/move_area/root=device:[directory.move]/nolog mf_personnel
%RMU-I-AIJRSTAVL, 1 after-image journal available for use
%RMU-I-AIJRSTMOD, 1 after-image journal marked as "modified"
%RMU-I-AIJISON, after-image journaling has been enabled
%RMU-W-DOFULLBCK, full database backup should be done to ensure future recovery
```

The following example shows the corrected behavior where the original state of AFTER IMAGE JOURNALING is restored (in this case AIJ is disabled both before and after the move).

```
$rmu/set after_journal/add=(name=aij1, file=aij1) mf_personnel
%RMU-W-AIJDEVDIR, AIJ filename "AIJ1" does not include a device/directory
%RMU-I-LOGCREAIJ, created after-image journal file DEVICE:[DIRECTORY]AIJ1.AIJ;1
%RMU-I-LOGMODSTR,     added after-image journal definition "AIJ1"
%RMU-W-DOENBLAIJ, after-image journaling must be enabled to ensure recovery
$rmu/set after_journal/enable mf_personnel
```

```
%RMU-I-LOGMODSTR,     activated after-image journal "AIJ1"
%RMU-I-LOGMODFLG,     enabled after-image journaling
%RMU-W-DOFULLBCK, full database backup should be done to ensure future recovery
$rmu/set after_journal/disable mf_personnel
%RMU-I-LOGMODFLG,     disabled after-image journaling
$rmu/move_area/root=device:[directory.move]/nolog mf_personnel
%RMU-I-AIJRSTAVL, 1 after-image journal available for use
%RMU-I-AIJRSTMOD, 1 after-image journal marked as "modified"
%RMU-I-AIJISOFF, after-image journaling has been disabled
%RMU-W-DOFULLBCK, full database backup should be done to ensure future recovery
```

A workaround for this problem is to specify RMU/MOVE_AREA/NOAFTER_JOURNAL to disable AFTER IMAGE JOURNALING or RMU/MOVE_AREA/AFTER_JOURNAL to enable it.

This problem has been corrected in Oracle Rdb Release 7.2.0.1.

4.3.6 Threshold for RMU System Area First Backup and Restore Now 100 Areas

Bug 5024112

VMS process virtual address space could be used up since a large number of restore writer threads were waiting for the system area to be restored so that each writer thread could recreate the ABM pages for the storage area it had just finished restoring. This caused new writer threads to be created for other storage areas instead of reusing existing writer threads that had completely finished restoring a storage area. Each writer thread that was created instead of reused was an additional drain on system resources. This was previously fixed by making sure the system area got backed up and restored first if there were 1000 or more database storage areas. This assures that more writer threads will be reused and not created. However, this problem has occurred for databases with fewer than 1000 storage areas. Therefore we have changed this so that the system area gets backed up and restored first if there are 100 or more database storage areas.

The following example shows this problem occurring with a database containing over 100 storage areas.

```
$ rmu/restore/nolog/nocdd/noafter manyareas.rbf
%COSI-F-VASFULL, virtual address space full
-SYSTEM-F-ILLPAGCNT, illegal page count parameter
```

This problem has been corrected in Oracle Rdb Release 7.2.0.1.

4.3.7 Incorrect NOSEQROW Diagnostic from RMU Verify

Bug 5056475

In prior versions of Oracle Rdb, RMU Verify would report an unexpected and erroneous NOSEQROW error when the database contained any PROFILE entry. If the database uses SECURITY CHECKING IS INTERNAL then any USER, ROLE or PROFILE would cause this error from RMU Verify.

```
$ RMU /VERIFY /ROOT /NOLOG /TRANSACTION_TYPE=PROTECTED MF_PERSONNEL
%RMU-E-NOSEQROW, sequence id 1 has an entry in the root file but no row in
RDB$SEQUENCES
```

This problem has been corrected in Oracle Rdb Release 7.2.0.1. RMU Verify has been corrected to include Rdb system−generated sequences in its checking.

4.3.8 RMU Load No longer Works With the USER$ Table

Bug 5056799

OCI Services for Rdb was recently enhanced to hide the Oracle data dictionary tables so that queries, such as SHOW TABLE, would only show those tables of interest to the application programmer.

Unfortunately, this change had a side effect that impacts a recommended migration of the USER$ table contents to different databases, or to a database re−prepared using the RDB_NATCONN71.COM procedure. Specifically, it was recommended that RMU/UNLOAD be used for the USER$ table and then RMU/LOAD used to reload the data into this special table. The RMU/LOAD no longer works on the data dictionary tables after upgrading to SQL/Services Release 7.1.6 or SQL/Services Release 7.2.

The following example shows the error reported by RMU. This error indicates that RMU is ignoring the system tables.

```
$ RMU/LOAD MF_PERSONNEL USER$ USER.UNL
%RMU−F−RELNOTFND, Relation (USER$) not found
%RMU−I−DATRECSTO,   0 data records stored.
%RMU−F−FTL_LOAD, Fatal error for LOAD operation at 23−FEB−2006
```

This problem has been corrected in Oracle Rdb Release 7.2.0.1. RMU Load has been modified to allow loading into the data dictionary tables provided by SQL/Services OCI Services. Oracle recommends that care be taken when modifying these metadata tables in the database because incorrect information could cause OCI Services for Rdb to behave inconsistently.

4.3.9 ACCVIO When Using RMU/BACKUP/PLAN

Bug 5004913

When using RMU/BACKUP/PLAN, the executor node exited with an ACCVIO error. The ACCVIO did not happen without /PLAN.

This problem occurred due to an uninitialized queue header in the /PLAN code path. This problem has been corrected in Oracle Rdb Release 7.2.0.1.

4.3.10 ENDOFFILE When Using RMU/RESTORE/LIBRARY

Bug 5004913

When using RMU/RESTORE/LIBRARY the command failed with:

```
%RMU−F−READERR, error reading ...
−COSI−W−ENDOFFILE, end of file
```

This problem occurred due to RMU/BACKUP/LIBRARY and RMU/RESTORE/LIBRARY using different fixed block size values.

Now the user can specify the block size value as with any other RMU/BACKUP (restrictions apply). The RMU/RESTORE/LIBRARY command now obtains the block size value from the save set.

This problem has been corrected in Oracle Rdb Release 7.2.0.1.

4.3.11 %MTH−F−SQUROONEG Error from RMU Workload Collect

Bug 4106407

Running RMU/COLLECT OPTIMIZER/STAT=WORKLOAD fails with the following error:

```
%MTH−F−SQUROONEG, square root of negative value
  user PC 00000003
  %RMU−F−FATALOSI, Fatal error from the Operating System Interface.
  %RMU−F−FTL_ANA, Fatal error for ANALYZE operation at 19−DEC−2005 05:29:36.83
```

Currently the maximum cardinality is limited to 249.95 million rows in RMU to collect the workload statistics for a given table. Customer's tables have cardinalities of 345 million, 548 million, 671 million and 1.341 billion.

A correction has been made to print out an informational message if the table exceeds the cardinality limit. A plan to allow a higher limit will be considered in a future release.

This problem has been corrected in Oracle Rdb Release 7.2.0.1.

4.4 LogMiner Errors Fixed

4.4.1 Incorrect %RMU−W−RECVERDIF When Using LogMiner with Uncompressed Table

Bug 4968838

In prior Oracle Rdb releases, the RMU /UNLOAD /AFTER_JOURNAL command would incorrectly process some combinations of storage map information and would be unable to extract data.

For example, the following table and storage map definition could lead to the RMU /UNLOAD /AFTER_JOURNAL command failing due to the DISABLE COMPRESSION attribute being incorrectly evaluated:

```
CREATE TABLE N2 (I1 INT);
CREATE STORAGE MAP N2 FOR N2 DISABLE COMPRESSION;
```

When this table was specified during an RMU /UNLOAD /AFTER_JOURNAL operation, an error similar to the following could be signaled:

```
%RMU−W−RECVERDIF, Record at dbkey 47:954:0 in table "N2"
version 12288 does not match current version
```

This problem has been corrected in Oracle Rdb Release 7.2.0.1.

Chapter 5
Software Errors Fixed in Oracle Rdb Release 7.2

This chapter describes software errors that are fixed by Oracle Rdb Release 7.2.

5.1 Software Errors Fixed That Apply to All Interfaces

5.1.1 Various Sequence Generation Problems Fixed

Several problems with sequence value generation have been corrected in this release. These problems occur at the extreme end of value ranges.

1. In some cases, when the end of the range of values is reached and the remaining values do not fill a cache, Rdb would discard the remaining cached values so that the last few values were never used. The cache size is defined by the CACHE clause of CREATE/ALTER SEQUENCE, or the SET FLAGS 'SEQ_CACHE(n)' statement. The following shows an example of this issue.

```
SQL> show sequence s3;
     S3
 Sequence Id: 1
 Initial Value: 29
 Minimum Value: 20
 Maximum Value: 30
 Next Sequence Value: 29
 Increment by: 1
 Cache Size: 5
 No Order
 Cycle
 No Randomize
 Wait
SQL> select s3.nextval from employees limit to 13 rows;

                    29
                    30
                    20
                    21
                    22
                    23
                    24
                    20
                    21
                    22
                    23
                    24
                    25
13 rows selected
```

   Note here that the sequence never returns the values 25, 26, 27, 28, 29, 30 after the initial cycle.
2. When a sequence approached the largest positive BIGINT value or the smallest negative BIGINT value, it was possible that an integer overflow could occur and cause the sequence to return incorrect values.
   This example should only generate negative values for the sequence.

```
SQL> show sequence sss;
     SSS
 Sequence Id: 1
 Initial Value: -9223372036854775807
```

```
 Minimum Value: -9223372036854775808
 Maximum Value: -1
 Next Sequence Value: -9223372036854775807
 Increment by: -1
 Cache Size: 20
 No Order
 No Cycle
 No Randomize
 Wait
 Comment:       Return only negative values
SQL> select sss.nextval from rdb$relations limit to 8 rows;

 -9223372036854775807
 -9223372036854775808
  9223372036854775807
  9223372036854775806
  9223372036854775805
  9223372036854775804
  9223372036854775803
  9223372036854775802
8 rows selected
SQL>
```

3. In prior releases, the upper and lower values of the sequence were restricted by subtracting the cache size from the maximum value and adding the cache size to the minimum value in an attempt to avoid overflow errors. This is no longer performed. Therefore, new CREATE CACHE or ALTER CACHE statements that use NOMAXVALUE, MAXVALUE BIGINT, NOMINVALUE, or MINVALUE BIGINT will now default to the largest positive BIGINT value minus one, or the smallest negative BIGINT value plus one. These values (−9223372036854775808, 9223372036854775807) are reserved for use by the sequence generator and may not be specified as a value for the START WITH clause of CREATE SEQUENCE or the RESTART WITH clause of ALTER SEQUENCE. SHOW SEQUENCE may now display a changed value for new sequences.

These problems have been corrected in Oracle Rdb Release 7.2.

5.1.2 Various Errors Using Read Only Areas with Global Buffers

Bug 4630467

Various problems could occur when accessing read only storage areas when global buffers were enabled and multiple processes were accessing the same pages at the same time. For example, bugchecks with exceptions similar to the following could occur:

```
***** Exception at 00EA6948 : RDMS$$EXE_NEXT + 000009D8
%COSI-F-BUGCHECK, internal consistency failure
```

This problem would occur because the buffer was not properly interlocked, allowing multiple users to read data into the buffer at the same time. The contents of the buffer would not be consistent until all I/Os had completed. For very brief instances, the buffer could contain some zeros while it was being filled in by the I/O request. Some users could be accessing the buffer while it contained the transient zeros, leading to various failures or incorrect results.

This problem can be avoided by changing the storage area to be READ WRITE or by disabling global buffers.

This problem has been corrected in Oracle Rdb Release 7.2.

5.1.3 Enhancement for HASH ORDERED Index of BIGINT Column

Bug 3103900

Previously, a HASH ORDERED index of a BIGINT column only considered the least significant 32 bits of the 64 bit value. This lead to unexpected record placement within the storage area. This, in some cases, could lead to severe hash collision problems resulting in excessive I/O and fragmentation issues.

This problem has been corrected in Oracle Rdb Release 7.2. For HASH ORDERED indexes created with Oracle Rdb Release 7.2, an improved algorithm is used when the data type is a BIGINT column. This algorithm considers the full 64−bit value for the hash function. Existing indexes or those indexes not using a BIGINT column retain use of the prior algorithm (utilizing only 32 bits of a 64 bit value).

Customers with databases using HASH ORDERED indexes of a BIGINT column may wish to consider dropping and recreating the index(es) to take advantage of the new algorithm. If the storage map for the table specifies placement via the effected index, then the table should also be reorganized using an unload and reload to ensure correct record placement.

5.1.4 RMU /SHOW LOCKS Limits Relaxed

Bug 3963053

Previously, the "/LOCK=" and "/PROCESS=" qualifiers of the "RMU /SHOW LOCKS" command were limited to 32 specified values.

This problem has been corrected in Oracle Rdb Release 7.2. The "/LOCK=" and "/PROCESS=" qualifiers of the "RMU /SHOW LOCKS" command now accept up to 256 values each.

5.2 SQL Errors Fixed

5.2.1 %SQL−F−UNSDATASS When ALIAS Defined With COMPILETIME PATHNAME

Bug 2315236

Column datatypes for columns defined with one of the ANSI date/time datatypes retrieved from CDD using a COMPILETIME PATHNAME were not correct. Specifically, they always appeared to be of datatype DATE VMS. This caused the generation of a %SQL−F−UNSDATASS during compilation when assignments were made between the affected columns and a properly typed host variable. The problem affected both SQL Module Language and the SQL Precompiler.

Note: You also must run CDD Release 7.2 for this Bug fix to be effective. Otherwise the behavior will be the same as before the fix. That is, SQL$PRE and SQL$MOD will still generate a %SQL−F−UNSDATASS error.

For example, consider the following database definition:

```
VMS> SQL$
SQL> CREATE DATABASE FILENAME TEST_DB PATHNAME TEST_DB;
SQL> CREATE TABLE T_TABLE (T_STAMP TIMESTAMP(2));
SQL> COMMIT;
SQL> EXIT;
```

The following precompiled COBOL program uses a COMPILETIME PATHNAME to retrieve metadata for the T_STAMP column of the T_STAMP table. Note that it also uses a SQL INCLUDE to generate a record named "T_TABLE" from the CDD. This record has the T_STAMP field properly typed. But the column datatype of the T_STAMP column was improperly typed.

```
IDENTIFICATION DIVISION.
PROGRAM−ID.          DATE_TIME_TEST.
AUTHOR.             TEST.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING−STORAGE SECTION.
01  SQLCODE                     PIC S9(9) COMP.
EXEC SQL
  INCLUDE FROM DICTIONARY 'TEST_DB.RDB$RELATIONS.T_TABLE'
END_EXEC.

EXEC SQL DECLARE ALIAS COMPILETIME PATHNAME TEST_DB
                       RUNTIME FILENAME TEST_DB
END_EXEC.

EXEC SQL
   DECLARE TIMESTAMP_TABLE_CURSOR READ ONLY CURSOR FOR
   SELECT * FROM T_TABLE
END_EXEC.

PROCEDURE DIVISION.
00000_ROOT_MODULE.
```

```
EXEC SQL
    OPEN TIMESTAMP_TABLE_CURSOR
END_EXEC.

EXEC SQL
    FETCH TIMESTAMP_TABLE_CURSOR INTO :T_TABLE
END_EXEC.

EXEC SQL
    CLOSE TIMESTAMP_TABLE_CURSOR
END_EXEC.

    STOP RUN.
```

When the program above was compiled, it would generate a *%SQL−F−UNSDATASS* as follows:

```
VMS> SQL$PRE/COBOL TEST_PROGRAM.SCO
    FETCH TIMESTAMP_TABLE_CURSOR INTO :T_TABLE
                                      1
%SQL−F−UNSDATASS, (1) Unsupported date/time assignment from T_STAMP to
T_STAMP IN T_TABLE
```

As a workaround, use COMPILETIME FILENAME.

This problem has been corrected in Oracle Rdb Release 7.2.

5.2.2 SQL Module Language /PROTOTYPES Now Generates INT64 for BIGINT Parameters

Bug 4721570

In prior releases of Oracle Rdb, the SQL Module Language qualifier /PROTOTYPES (or /C_PROTOTYPES) would generate an interface definition that used pointer to long type for the SQL procedure BIGINT parameters. This caused warnings from the C compiler as shown in the following example.

```
MYPROC (sqlstate, &s, &i, &l);
.......................^
%CC−W−PTRMISMATCH, In this statement, the referenced type of the pointer
value "&l" is "__int64", which is not compatible with "long"
at line number 11 in file USER2:[TESTING]SAMPLE.C;1
```

This problem has been corrected in Oracle Rdb Release 7.2. SQL Module Language now generates int64 as the type in the prototype declaration. Applications should include ints.h in their applications and also use int64 when fetching BIGINT data. This is shown in this cut down example.

```
#include <ints.h>
#include <user2:[testing]mod.h>
#include <stdlib.h>

void main ()
{
char sqlstate[5] = "00000";
short s = 0;
int i = 0;
int64 l = 0;
MYPROC (sqlstate, &s, &i, &l);
```

```
exit (0);
}
```

5.2.3 Unexpected Column Ordering Generated by ALTER TABLE ... ALTER COLUMN Statements

Bug 4624762

In prior versions of Oracle Rdb when multiple ALTER COLUMN { BEFORE | AFTER } COLUMN clauses were used in an ALTER TABLE, the resulting column order would be different from that created by multiple ALTER TABLE statements which included just one ALTER COLUMN clause.

The following example shows the problem. Both sets of ALTER TABLE statements should result in the same column ordering.

```
SQL> create table PERSON
cont>     (address char(40)
cont>     ,last_name char(30)
cont>     ,social_security_number integer
cont>     ,first_name char(30)
cont>     );
SQL> commit;
SQL>
SQL> show table (column) PERSON;
Information for table PERSON

Columns for table PERSON:
Column Name                    Data Type        Domain
-----------                    ---------        ------
ADDRESS                        CHAR(40)
LAST_NAME                      CHAR(30)
SOCIAL_SECURITY_NUMBER         INTEGER
FIRST_NAME                     CHAR(30)

SQL>
SQL> alter table PERSON
cont>     alter column first_name
cont>         before column last_name;
SQL> alter table PERSON
cont>     alter column social_security_number
cont>         after column address;
SQL>
SQL> show table (column) PERSON;
Information for table PERSON

Columns for table PERSON:
Column Name                    Data Type        Domain
-----------                    ---------        ------
ADDRESS                        CHAR(40)
SOCIAL_SECURITY_NUMBER         INTEGER
FIRST_NAME                     CHAR(30)
LAST_NAME                      CHAR(30)

SQL>
SQL> rollback;
SQL>
SQL> alter table PERSON
```

```
cont>     alter column first_name
cont>         before column last_name
cont>     alter column social_security_number
cont>         after column address;
SQL>
SQL> show table (column) PERSON;
Information for table PERSON

Columns for table PERSON:
Column Name                    Data Type        Domain
-----------                    ---------        ------
ADDRESS                        CHAR(40)
FIRST_NAME                     CHAR(30)
SOCIAL_SECURITY_NUMBER         INTEGER
LAST_NAME                      CHAR(30)

SQL>
SQL> rollback;
```

This problem has been corrected in Oracle Rdb Release 7.2. Oracle Rdb now preserves the relative ordering of columns when multiple ALTER COLUMN clauses are used.

5.3 RMU Errors Fixed

5.3.1 RMU /UNLOAD Qualifier /REOPEN_COUNT

Bug 4246050

Previously, RMU /UNLOAD would write a single output file for all records in the table/view being unloaded. In some cases, this single output file would become difficult to manipulate.

This problem has been corrected in Oracle Rdb Release 7.2. The "/REOPEN_COUNT=n" qualifier allows one to specify how many records will be written to an output file. The output file will be re−created (ie, a new version of the file created) when the record count reaches the specified value. The "/REOPEN_COUNT=n" qualifier is only valid when used with the "/RECORD_DEFINITION" or "/RMS_RECORD_DEF" qualifiers.

5.3.2 RMU /LOAD /STATISTICS=ON_COMMIT

Previously, the "/STATISTICS=ON_COMMIT" qualifier would have no effect on the RMU /LOAD operations; the qualifier was ignored. The "/STATISTICS=INTERVAL=n" qualifier did function correctly.

This problem has been corrected in Oracle Rdb Release 7.2. When the STATISTICS=(ON_COMMIT) qualifier is specified, Oracle RMU now correctly prints statistics each time a transaction is committed.

Chapter 6
Enhancements And Changes Provided in Oracle Rdb Release 7.2.1.0

6.1 Enhancements And Changes Provided in Oracle Rdb Release 7.2.1.0

6.1.1 New Columns in Information Table RDB$JOURNALS

Bug 5401232

Columns RDB$SEQUENCE_NUMBER and RDB$STATE have been added to the Information Table RDB$JOURNALS. These contain the current AIJ sequence number and the STATE (either "Current" or "Latent") for the AIJ file.

```
SQL> select RDB$SEQUENCE_NUMBER, RDB$STATE from RDB$JOURNALS;
 RDB$SEQUENCE_NUMBER   RDB$STATE
                  -1   Latent
                  -1   Latent
                   3   Current
3 rows selected
```

To "upgrade" an existing database, which already contains the Information Table RDB$JOURNALS, simply drop the table and re−run SQL$SAMPLE:INFO_TABLES.SQL. This will create a new and complete RDB$JOURNALS table (any errors concerning the existence of any other Information Tables can be safely ignored). Or, drop the Information Table and recreate the Information Table with the required columns (see SQL$SAMPLE:INFO_TABLES.SQL for the available columns).

After installing Rdb Release 7.2.1, any database which contains an RDB$JOURNALS Information Table will continue to function as before but the new columns will not be visible. Also, previous versions of INFO_TABLES.SQL will still function as before but, again, the new columns will not be visible.

New column definitions for RDB$JOURNALS:

```
Column Name           Date Type   Domain
-----------           ---------   ------
RDB$SEQUENCE_NUMBER   integer     RDB$COUNTER
RDB$STATE             char(31)    RDB$USAGE
```

6.1.2 Oracle Rdb Release 7.2.x.x New Features Document Added

A new document has been created which contains all of the New Features Chapters from all previous Rdb 7.2 Release Notes. This document will be included in saveset A of the Rdb kit. It is called RDB_NEWFEATURES_72xx and will be available in postscript, text and PDF format. This will provide customers with one document to reference to find out about all new features that have been added to the Rdb 7.2 releases.

6.1.3 Hot Standby Status Symbols From RMU /SHOW AFTER_JOURNAL /BACKUP_CONTEXT

Additional DCL symbols indicating the Hot Standby replication state are now created by the RMU /SHOW AFTER_JOURNAL /BACKUP_CONTEXT command.

The symbol names are listed below:

- RDM$HOT_STANDBY_STATE − Contains the current replication state. Possible state strings and the description of each state are listed below:
  - "Inactive" − Inactive
  - "DB_Bind" − Binding to database
  - "Net_Bind" − Binding to network
  - "Restart" − Replication restart activity
  - "Connecting" − Waiting for LCS to connect
  - "DB_Synch" − Database synchronization
  - "Activating" − LSS server activation
  - "SyncCmpltn" − LRS synchronization redo completion
  - "Active" − Database replication
  - "Completion" − Replication completion
  - "Shutdown" − Replication cleanup
  - "Net_Unbind" − Unbinding from network
  - "Recovery" − Unbinding from database
  - "Unknown" − Unknown state or unable to determine state
- RDM$HOT_STANDBY_SYNC_MODE − Contains the current replication synchronization mode when replication is active. Possible synchronization mode strings are listed below:
  - "Cold"
  - "Warm"
  - "Hot"
  - "Commit"
  - "Unknown"

6.1.4 RMU BACKUP, COPY, MOVE /THREADS=n New Qualifier

A new qualifier has been added to allow the user to better control the system load created by a backup, copy or move operation. The new qualifier allows the user to specify the number of threads to be used by RMU.

RMU creates so called internal 'threads' of execution to read data from one specific storage area. Threads run quasi−parallel within the process executing the RMU image. Each thread generates its own I/O load and consumes resources like virtual address space and process quotas (e.g. FILLM, BYTLM). The more threads, the more I/Os can be generated at one point in time and the more resources are needed to accomplish the same task.

Performance increases with more threads due to parallel activities which keep disk drives busier. However, at a certain number of threads, performance suffers because the disk I/O subsystem is saturated and I/O queues build up for the disk drives. Also the extra CPU time for additional thread scheduling overhead reduces the overall performance. Typically 2−5 threads per input disk drive are sufficient to drive the disk I/O subsystem

at its optimum. However, some controllers may be able to handle the I/O load of more threads, e.g. disk controllers with RAID sets and extra cache memory.

In a COPY or MOVE operation, one thread moves the data of one storage area at a time. If there are more storage areas to be moved than there are threads, then the next idle thread takes on the next storage area. Storage areas are moved in order of the area size, largest areas first. This optimizes the overall elapsed time by allowing other threads to move smaller areas while an earlier thread is still working on a large area. If no threads qualifier is specified, then 10 threads are created by default. The minimum is 1 thread and the maximum is the number of storage areas to be copied or moved. If the user specifies a value larger than the number of storage areas, then RMU silently limits the number of threads to the number of storage areas.

In a BACKUP operation, one writer thread is created per output stream. An output stream can be either a tape drive, a disk file or a media library manager stream. In addition, RMU creates a number of reader threads and their number can be specified. RMU assigns a subset of reader threads to writer threads. RMU calculates the assignment so that roughly the same amount of data is assigned to each output stream. By default, five reader threads are created for each writer thread. If the user has specified the number of threads, then this number is used to create the reader thread pool. RMU always limits the number of reader threads to the number of storage areas. A threads number of 0 causes RMU to create one thread per storage area which start to run all in parallel immediately. Even though this may sound like a good idea to improve performance, this approach causes performance to suffer for databases with a larger number (>10) of storage areas. For a very large number of storage areas (>800), this fails due to hard limitations in system resources like virtual address space.

For a COPY or MOVE operation, you can specify a threads number as low as 1. Using a threads number of 1 generates the smallest system load in terms of working set usage and disk I/O load. Disk I/O subsystems most likely can handle higher I/O loads. Using a slightly larger value than 1 typically results in faster execution time.

For a BACKUP operation, the smallest threads number you can specify is the number of output streams. This gurantees that each writer thread has at least one reader thread assigned to it and does not produce an empty save set. Using a threads number equal to the number of output streams generates the smallest system load in terms of working set usage and disk I/O load. Disk I/O subsystems most likely can handle higher I/O loads. Using a slightly larger value than the number of output streams (assigning more reader threads to a writer thread), typically results in faster execution time.

The old READER_THREAD_RATIO qualifier has been deprecated but is still accepted and works exactly the same as in previous versions.

Examples using the /THREADS qualifier:

```
Copying one storage area at a time:

$ RMU /COPY /THREADS=1 /LOG FOO BCK
%RMU-I-MOVTXT_04, Starting move of storage area ...
%RMU-I-MOVTXT_01, Completed move of storage area ...
%RMU-I-MOVTXT_05, Moved snapshot area file ...
%RMU-I-MOVTXT_04, Starting move of storage area ...
%RMU-I-MOVTXT_01, Completed move of storage area ...
%RMU-I-MOVTXT_05, Moved snapshot area file ...
    .
    .
    .
```

```
Copying three storage areas in parallel:

$ RMU /COPY /THREADS=3 /LOG FOO BCK
%RMU-I-MOVTXT_04, Starting move of storage area ...
%RMU-I-MOVTXT_04, Starting move of storage area ...
%RMU-I-MOVTXT_04, Starting move of storage area ...
%RMU-I-MOVTXT_01, Completed move of storage area ...
%RMU-I-MOVTXT_05, Moved snapshot area file ...
%RMU-I-MOVTXT_04, Starting move of storage area ...
%RMU-I-MOVTXT_01, Completed move of storage area ...
%RMU-I-MOVTXT_05, Moved snapshot area file ...
    .
    .
    .
```

6.1.5 Concealed Logical Names Defined in LNM$SYSCLUSTER_TABLE Table Allowed

Previously, many uses of concealed logical device names were required to be defined in the LNM$SYSTEM_TABLE logical name table. This requirement is in place to ensure that various components of the database system running in separate process contexts would all have access to the same logical name definitions. Uses of concealed logical device names that were not defined in the LNM$SYSTEM_TABLE could result in a *COSI−F−NOTSYSCONCEAL "non−system concealed device name in filename" status*.

This restriction has been somewhat relaxed. While all processes using a database still require access to the same logical name definitions, this can now be accomplished by using the LNM$SYSTEM_TABLE logical name table or the LNM$SYSCLUSTER_TABLE logical name table (which represents a cluster−wide resource). Note, however, that it is strongly recommended that concealed logical device names not be defined in both tables at the same time on any cluster node as this can lead to unpredictable results possibly leading to database corruption or instability.

6.1.6 Support for GNAT Ada on Alpha and Itanium

Support has been added to Precompiled SQL and SQL Module Language for the Ada Core GNAT Ada compiler. This support allows SQL$PRE/ADA compilations to target the GNAT Ada compiler and facilitates interfacing SQL Module Language modules to GNAT Ada programs. For migrating existing applications from DEC Ada to GNAT Ada, in most cases the only changes needed are those required by the different rules of the two language variants. The most significant changes for most DEC Ada applications will be because GNAT Ada requires a source file to contain a single "compilation unit" which means a single package specification or a single package body. Files containing package specifications and bodies must use the suffixes .ADS and .ADB, respectively.

GNAT Ada uses a more Unix−like "compilation environment" in contrast to the Ada Development Library approach of DEC Ada. It consists of the following three steps: GNAT COMPILE which produces object files and .ALI files (Ada Library Information); GNAT BIND which checks consistency, determines the order of elaboration, and generates a main program which incorporates that elaboration; and GNAT LINK which compiles the main program from GNAT BIND, builds a set of linker options, and calls the OpenVMS link utility to produce an executable program. There is also a utility called GNAT MAKE which folds these steps together, including detecting obsolete programs and recompiling them. In most cases, Precompiled SQL applications and applications which call SQL Module Language modules can be built using GNAT MAKE provided that the .SQLADA and .SQLMOD source code files are compiled with SQL$PRE or SQL$MOD

beforehand.

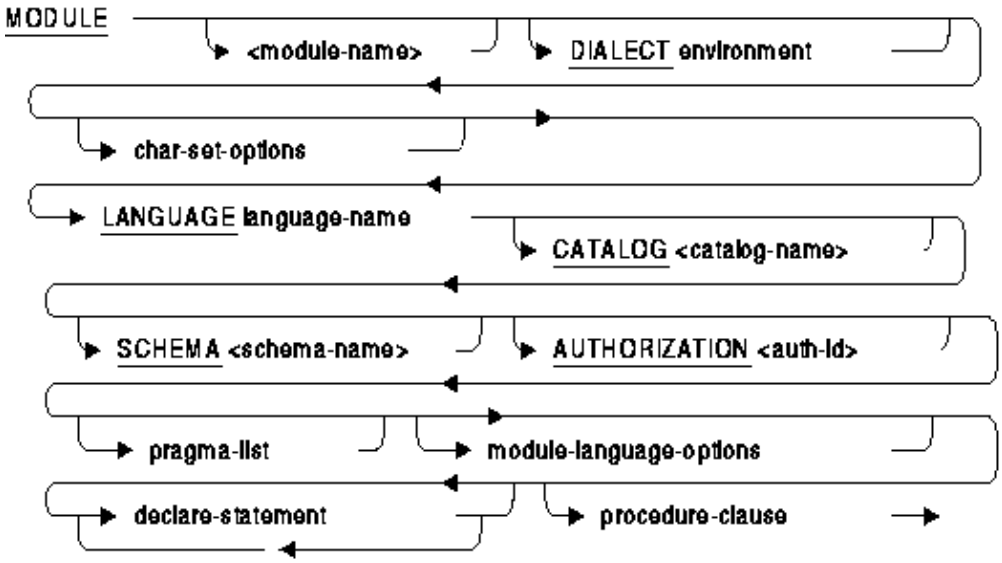
For information about GNAT Ada development see the Ada Core documentation for GNAT Ada on OpenVMS. The following specific Ada Core documents are pertinent:

- GNAT Pro User's Guide − OpenVMS − GNAT Pro Ada 95 Compiler
- GNAT Pro Reference Manual − GNAT Pro Ada 95 Compiler

The minimum supported version of the Ada Core GNAT Ada compilers are as follows: for Alpha, 5.04a1 and for Itanium, 5.04a.

On Itanium, GNAT Ada is the only option for Ada development since DEC Ada is not supported by HP on Itanium. On Alpha, either DEC Ada or GNAT Ada may be used. For SQL$PRE, this choice is determined by values added to the /ADA qualifier as follows: /ADA={DEC_ADA,GNAT_ADA} where DEC_ADA is the default.

For SQL Module Language on Alpha, there are two means of specifying which Ada compiler is the target. First, there is a new clause in the module header called "PRAGMA" which can have valid keywords of GNAT_ADA and DEC_ADA (but only one of them). In the future, additional keywords may be added to the PRAGMA clause for other purposes. The pragma clause appears in the module header according to the following syntax:



pragma-list =

The following example shows the use of a PRAGMA clause in a module header to specify that the target is GNAT Ada:

```
MODULE          MY_MODULE
DIALECT         SQL99
LANGUAGE        ADA
AUTHORIZATION   SAMPLE_USER
PRAGMA          (GNAT_ADA)
ALIAS           RDB$DBHANDLE
PARAMETER       COLONS
```

The second method of specifying the target compiler is a new SQL$MOD qualifier /PRAGMA={GNAT_ADA,DEC_ADA} on the command line. A PRAGMA clause in the code takes precedence over the qualifier.

6.1.6.1 Pragma EXTEND_SYSTEM

SQL depends on certain types which are defined in package SYSTEM in DEC Ada. Many of these types are not in the Ada Core implementation. A special pragma exists in the Ada Core implementation which allows use of these types and their associated functions. It is as follows:

```
pragma EXTEND_SYSTEM (AUX_DEC);
```

This pragma can be added to the GNAT Ada compilation configuration file (GNAT.ADC) in your compilation directory and will automatically be applied to all compilations. See the Ada Core documentation for more information about the GNAT Ada compilation environment and the use of the GNAT.ADC file.

6.1.6.2 SQL_STANDARD Package

For DEC Ada, the package SQL_STANDARD is stored in a file named SQL$STANDARD.ADA which is placed in SYS$LIBRARY by the Rdb installation procedure. In order to conform to GNAT Ada naming conventions, a new file has been created to contain the SQL_STANDARD package. This file is SQL_STANDARD.ADS and is placed in SYS$LIBRARY by the Rdb install procedures.

6.1.6.3 GNAT Ada Type Differences

With GNAT Ada, the default address size for the SYSTEM.ADDRESS type is 64 bits. All SQL routines currently use 32 bit addresses. Accordingly, the definition of the Ada SQLVAR_REC record (which is used in the SQLDA) has been changed so the SQLDATA and SQLIND components are SYSTEM.SHORT_ADDRESS in lieu of SYSTEM.ADDRESS.

Other relevant GNAT Ada type differences have to do with floating point datatypes (corresponding to the SQL REAL and DOUBLE PRECISION datatypes). Instead of providing types for explicit floating point representation in package SYSTEM, GNAT Ada provides pragmas to specify the floating point representation for the types in package STANDARD. These pragmas are Float_Representation and Long_Float. Float_Representation allows you to specify IEEE_Float or VAX_Float. If you specify VAX_Float, pragma Long_Float allows you to specify D_Float or G_Float. The package STANDARD types which are relevant to SQL and affected be these pragmas are SINGLE_FLOAT and DOUBLE_FLOAT. See the Ada Core documentation for more information about the GNAT Ada floating point pragmas.

6.1.6.4 SQL Module Language

SQL$MOD generates and calls GNAT COMPILE to compile an Ada package specification which has the same name as the module and the suffix ".ADS". For the example module header above, the package specification file would be "MY_MODULE.ADS". GNAT COMPILE creates an object file which, continuing the example, would be named "MY_MODULE.OBJ". The SQL object file generated directly by SQL$MOD will, by default, have the same prefix as the .SQLMOD source file and a suffix of .OBJ or, if the "/OBJECT=" qualifier is used, the name specified by that qualifier. SQL$MOD will detect if this name duplicates the name of the object file out of the GNAT Ada compiler and, if so, will use a .SQL_OBJ suffix for its object file to avoid the name conflict. SQL$MOD also generates a VMS linker utility options file which allows GNAT LINK to link in the SQL object file to the application. This options file, named "module_name.OPT", also contains an entry for the SQL$USER library so that GNAT LINK will be able to resolve the references to it in the SQL$MOD−generated object file. SQL$MOD adds a "pragma Linker_Options" to the generated .ADS file to tell GNAT LINK about the generated .OPT file. GNAT LINK integrates the SQL$MOD−generated options file into the options file that it creates for the OpenVMS linker utility. This approach allows most GNAT Ada applications which call SQL Module Language modules to be built using the GNAT MAKE utility once the SQL$MOD compile is completed.

6.1.6.5 Precompiled SQL

Precompiled SQL generates several output files for a .SQLADA source file. One of these is the Ada source file which contains the Ada code in the original SQLADA file with the EXEC SQL statements translated into procedure calls. With DEC Ada, this file has the same name as the original SQLADA file but with a .ADA extension. When targeting GNAT Ada, this file has an extension of .ADB in order to conform to GNAT Ada naming conventions. SQL$PRE calls the GNAT compiler to compile the .ADB file into an object file with the .OBJ suffix and the .ALI file needed by the GNAT BIND and GNAT LINK commands. As with DEC Ada, SQL$PRE replaces the EXEC SQL statements with calls to routines in a SQL module. SQL$PRE produces an object file for the SQL module and an Ada package specification for it. As with DEC Ada, the default file name prefix for generated SQL module files is formed by prefixing "SQL_" on the original file name. The only difference is that the Ada package spec for the SQL module has the suffix ".ADS" in conformance with GNAT Ada conventions.

When targeting GNAT Ada, SQL$PRE automatically calls the GNAT Ada compiler with the generated Ada files just as it does for DEC Ada. Both the .ADB and .ADS files generated by SQL$PRE are compiled so that an executable can be built using the GNAT BIND and GNAT LINK commands. The compilation of the .ADS file results in an object file which would have the same name as the object file generated by SQL$PRE, that is: "SQL_module_name.OBJ". Accordingly, the object file generated by SQL$PRE is named: "SQL_module_name.SQL_OBJ". SQL$PRE generates a VMS linker utility options file which allows GNAT LINK to link in the .SQL_OBJ file to the application. This options file, named "SQL_module_name.OPT", also contains an entry for the SQL$USER library so that GNAT LINK will be able to resolve the references to it in the SQL$PRE−generated object file. A "pragma Linker_Options" is added to the .ADS file to tell GNAT LINK about the generated .OPT file.

For GNAT Ada, the declaration of RDB_MESSAGE_VECTOR as an object mapping to the RDB$MESSAGE_VECTOR PSECT has been moved to the .ADB file because the generated code will not link properly if this declaration is in a .ADS file.

The following example shows building a Precompiled SQL application using the GNAT Ada compiler. Note that on Itanium, the "=/GNAT_ADA" qualifier would be unnecessary because only GNAT Ada is supported on that platform. This example shows how to build and run the Ada version of the SQL_ALL_DATATYPES

application from SQL$SAMPLES.

```
$!
$! Set up GNAT Ada environment
$!
$ CREATE GNAT.ADC
pragma EXTEND_SYSTEM (AUX_DEC);
$ DEFINE ADA_INCLUDE_PATH SYS$LIBRARY
$ GNAT COMPILE SYS$LIBRARY:SQL_STANDARD.ADS
$!
$! Build SQL_ALL_DATATYPES
$!
$ SQL$PRE/ADA=GNAT_ADA SQL$SAMPLE:SQL_ALL_DATATYPES
$ GNAT MAKE SQL_ALL_DATATYPES
$!
$ RUN SQL_ALL_DATATYPES
```

6.1.7 Enhancement to SQLCA

The following enhancements have been made to the SQLCA with this release:

- The SQLCA field SQLERRD[0] is now updated with the statement type by the PREPARE statement for all dialects. These numeric codes are listed in the table below.
  In previous releases, SQLERRD[0] was set only for ORACLE LEVEL1 and ORACLE LEVEL2 dialects.
- If the statement being prepared is a SELECT statement containing an INTO clause, then SQLCA field SQLWARN6 will contain the character "I". Such singleton SELECT statements can be executed without using a cursor.

Table 6−1 SQLCA SQLERRD [0] Values

Symbolic Name+	Value	SQL Statement
	0	Statement is unknown
SQL_K_OCTRDB_CONNECT	−1	Rdb Connect
SQL_K_OCTRDB_ATTACH	−2	Rdb Attach
SQL_K_OCTRDB_DISCONNECT	−3	Rdb Disconnect
SQL_K_OCTRDB_CREATE_MODULE	−4	Rdb Create Module
SQL_K_OCTRDB_ALTER_MODULE	−5	Rdb Alter Module
SQL_K_OCTRDB_DROP_MODULE	−6	Rdb Drop Module
SQL_K_OCTRDB_CREATE_DOMAIN	−7	Rdb Create Domain
SQL_K_OCTRDB_ALTER_DOMAIN	−8	Rdb Alter Domain
SQL_K_OCTRDB_DROP_DOMAIN	−9	Rdb Drop Domain
SQL_K_OCTRDB_CREATE_CATALOG	−10	Rdb Create Catalog
SQL_K_OCTRDB_ALTER_CATALOG	−11	Rdb Alter Catalog
SQL_K_OCTRDB_DROP_CATALOG	−12	Rdb Drop Catalog
SQL_K_OCTRDB_ALTER_SCHEMA	−13	Rdb Alter Schema
SQL_K_OCTRDB_DROP_SCHEMA	−14	Rdb Drop Schema

SQL_K_OCTRDB_SET_SESSION	−15	Rdb Set Session Authorization
SQL_K_OCTCTB	1	create table
SQL_K_OCTINS	2	insert
SQL_K_OCTSEL	3	select
SQL_K_OCTCCL	4	create cluster
SQL_K_OCTACL	5	alter cluster
SQL_K_OCTUPD	6	update
SQL_K_OCTDEL	7	delete
SQL_K_OCTDCL	8	drop cluster
SQL_K_OCTCIX	9	create index
SQL_K_OCTDIX	10	drop index
SQL_K_OCTAIX	11	alter index
SQL_K_OCTDTB	12	drop table
SQL_K_OCTCSQ	13	create sequence
SQL_K_OCTASQ	14	alter sequence
SQL_K_OCTATB	15	alter table
SQL_K_OCTDSQ	16	drop sequence
SQL_K_OCTGRA	17	grant
SQL_K_OCTREV	18	revoke
SQL_K_OCTCSY	19	create synonym
SQL_K_OCTDSY	20	drop synonym
SQL_K_OCTCVW	21	create view
SQL_K_OCTDVW	22	drop view
SQL_K_OCTVIX	23	validate index
SQL_K_OCTCPR	24	create procedure
SQL_K_OCTAPR	25	alter procedure
SQL_K_OCTLTB	26	lock table
SQL_K_OCTNOP	27	no operation
SQL_K_OCTRNM	28	rename
SQL_K_OCTCMT	29	comment
SQL_K_OCTAUD	30	audit
SQL_K_OCTNOA	31	noaudit
SQL_K_OCTCED	32	create database link
SQL_K_OCTDED	33	drop database link
SQL_K_OCTCDB	34	create database
SQL_K_OCTADB	35	alter database
SQL_K_OCTCRS	36	create rollback segment
SQL_K_OCTARS	37	alter rollback segment
SQL_K_OCTDRS	38	drop rollback segment
SQL_K_OCTCTS	39	create tablespace
SQL_K_OCTATS	40	alter tablespace
SQL_K_OCTDTS	41	drop tablespace

SQL_K_OCTASE	42	alter session
SQL_K_OCTAUR	43	alter user
SQL_K_OCTCWK	44	commit
SQL_K_OCTROL	45	rollback
SQL_K_OCTSPT	46	savepoint
SQL_K_OCTPLS	47	pl/sql execute
SQL_K_OCTSET	48	set transaction
SQL_K_OCTASY	49	alter system switch log
SQL_K_OCTXPL	50	explain
SQL_K_OCTCUS	51	create user
SQL_K_OCTCRO	52	create role
SQL_K_OCTDUS	53	drop user
SQL_K_OCTDRO	54	drop role
SQL_K_OCTSER	55	set role
SQL_K_OCTCSC	56	create schema
SQL_K_OCTCCF	57	create control file
SQL_K_OCTATR	58	Alter tracing
SQL_K_OCTCTG	59	create trigger
SQL_K_OCTATG	60	alter trigger
SQL_K_OCTDTG	61	drop trigger
SQL_K_OCTANT	62	analyze table
SQL_K_OCTANI	63	analyze index
SQL_K_OCTANC	64	analyze cluster
SQL_K_OCTCPF	65	create profile
SQL_K_OCTDPF	66	drop profile
SQL_K_OCTAPF	67	alter profile
SQL_K_OCTDPR	68	drop procedure
SQL_K_OCTARC	70	alter resource cost
SQL_K_OCTCSL	71	create snapshot log
SQL_K_OCTASL	72	alter snapshot log
SQL_K_OCTDSL	73	drop snapshot log
SQL_K_OCTCSN	74	create snapshot
SQL_K_OCTASN	75	alter snapshot
SQL_K_OCTDSN	76	drop snapshot
SQL_K_OCTCTY	77	create type
SQL_K_OCTDTY	78	drop type
SQL_K_OCTARO	79	alter role
SQL_K_OCTATY	80	alter type
SQL_K_OCTCYB	81	create type body
SQL_K_OCTAYB	82	alter type body
SQL_K_OCTDYB	83	drop type body
SQL_K_OCTDLB	84	drop library

SQL_K_OCTTTB	85	truncate table
SQL_K_OCTTCL	86	truncate cluster
SQL_K_OCTCBM	87	create bitmapfile
SQL_K_OCTAVW	88	alter view
SQL_K_OCTDBM	89	drop bitmapfile
SQL_K_OCTSCO	90	set constraints
SQL_K_OCTCFN	91	create function
SQL_K_OCTAFN	92	alter function
SQL_K_OCTDFN	93	drop function
SQL_K_OCTCPK	94	create package
SQL_K_OCTAPK	95	alter package
SQL_K_OCTDPK	96	drop package
SQL_K_OCTCPB	97	create package body
SQL_K_OCTAPB	98	alter package body
SQL_K_OCTDPB	99	drop package body
SQL_K_OCTCDR	157	create directory
SQL_K_OCTDDR	158	drop directory
SQL_K_OCTCLB	159	create library
SQL_K_OCTCJV	160	create java
SQL_K_OCTAJV	161	alter java
SQL_K_OCTDJV	162	drop java
SQL_K_OCTCOP	163	create operator
SQL_K_OCTCIT	164	create indextype
SQL_K_OCTDIT	165	drop indextype
SQL_K_OCTAIT	166	reserver for alter indextype
SQL_K_OCTDOP	167	drop operator
SQL_K_OCTAST	168	associate statistics
SQL_K_OCTDST	169	disassociate statistics
SQL_K_OCTCAL	170	call method
SQL_K_OCTCSM	171	create summary
SQL_K_OCTASM	172	alter summary
SQL_K_OCTDSM	173	drop summary
SQL_K_OCTCDM	174	create dimension
SQL_K_OCTADM	175	alter dimension
SQL_K_OCTDDM	176	drop dimension
SQL_K_OCTCCT	177	create context
SQL_K_OCTDCT	178	drop context
SQL_K_OCTASO	179	alter outline
SQL_K_OCTCSO	180	create outline
SQL_K_OCTDSO	181	drop outline
SQL_K_OCTAOP	183	alter operator
SQL_K_OCTCEP	184	create encryption profile

SQL_K_OCTAEP	185	alter encryption profile
SQL_K_OCTDEP	186	drop encryption profile
SQL_K_OCTCSP	187	create spfile from pfile
SQL_K_OCTCPS	188	create pfile from spfile
SQL_K_OCTUPS	189	merge
SQL_K_OCTCPW	190	change password
SQL_K_OCTUJI	191	update join index
SQL_K_OCTASYN	192	alter synonym
SQL_K_OCTADG	193	alter disk group
SQL_K_OCTCDG	194	create disk group
SQL_K_OCTDDG	195	drop disk group
SQL_K_OCTALB	196	alter library
SQL_K_OCTPRB	197	purge user recyclebin
SQL_K_OCTPDB	198	purge dba recyclebin
SQL_K_OCTPTS	199	purge tablespace
SQL_K_OCTPTB	200	purge table
SQL_K_OCTPIX	201	purge index
SQL_K_OCTUDP	202	undrop object
SQL_K_OCTDDB	203	drop database
SQL_K_OCTFBD	204	flashback database
SQL_K_OCTFBT	205	flashback table

+ The positive values are defined for compatibility with Oracle 10g. Not all statements are supported by Oracle Rdb therefore not all values will appear in the SQLCA. Negative values are Oracle Rdb specific values.

6.1.8 File−System Caching Avoided for RMU /COPY, /MOVE, /BACKUP And /RESTORE, IO To Database

In order to reduce CPU consumption and XFC spinlock contention and to help avoid "thrashing" the file system cache and to streamline file read and write operations, caching by the operating system is disabled for various files and operations including:

- RMU /COPY
- RMU /MOVE
- RMU /BACKUP
- RMU /RESTORE
- Most Database Root File IO
- Most Database RUJ File IO
- Most Row−Cache Backing Store File IO
- Most Recovery Work File IO

Testing on various configurations indicates that, in general, avoiding the operating system's XFC cache for these database file IO operations results in better overall performance as balanced between CPU and IO costs.

6.1.9 RMU /BACKUP /COMPRESSION New Algorithm

The RMU /BACKUP /COMPRESSION feature has been enhanced to offer an additional compression algorithm. The ZLIB algorithm and software, developed by Jean−loup Gailly and Mark Adler, has been implemented for RMU /BACKUP /COMPRESS. This implementation generally uses the same or less CPU time and is generally more effective (compresses better) than either of the HUFFMAN or LZSS algorithms.

The /COMPRESSION qualifier accepts the following keywords:

- HUFFMAN − HUFFMAN encoding algorithm.
- LZSS − Lempel−Ziv algorithm.
- ZLIB=level − ZLIB algorithm. The "level" value is an integer between 1 and 9 specifying the relative compression level with one being the least amount of compression and nine being the greatest amount of compression. Higher levels of the compression use increased CPU time while generally providing better compression. The default compression level of 6 is a balance between compression effectiveness and CPU consumption.

If you specify the /COMPRESSION qualifier without a value, the default is /COMPRESSION=ZLIB=6.

Here are examples using the /COMPRESS qualifier. Note that if "/LOG=FULL" is specified, data compression statistics information is displayed.

```
$ RMU /BACKUP /COMPRESS /NOLOG FOO BCK
$ RMU /BACKUP /COMPRESS=ZLIB:9 /LOG=FULL FOO BCK
    .
    .
    .
BACKUP summary statistics:
        Data compressed by 53% (9791 KB in/4650 KB out)
```

Older Oracle Rdb 7.2 Releases and Compressed RBF Files

Prior releases of Oracle Rdb are unable to read RBF files compressed with the ZLIB algorithm. In order to read compressed backups with Oracle Rdb Release 7.2 prior to V7.2.1, they must be made with /COMPRESSION=LZSS or /COMPRESSION=HUFFMAN explicitly specified (because the default compression algorithm has been changed from LZSS to ZLIB). Oracle Rdb Release 7.2.1 is able to read compressed backups using the LZSS or HUFFMAN algorithms made with prior releases.

Compression Effectiveness Varies

The actual amount of compression for any algorithm is strongly dependent on the actual data being compressed. Some database content may compress quite well and other content may compress not at all and may actually result in expansion of the output.

When using the /ENCRYPT and /COMPRESS features together, data is first compressed and then encrypted. This provides effective compression as well as effective encryption.

6.1.10 Enhancements for Compression Support in RMU Unload and Load

Bugs 690179 and 675012

This release of Oracle Rdb introduces support for compression to RMU Unload, RMU Load and RMU Dump Export.

Data compression is applied to the user data unloaded to the internal (interchange) format file. Table rows, null byte vector and LIST OF BYTE VARYING data is compressed using either the LZW (Lempel−Ziv−Welch) technique or the ZLIB algorithm developed by Jean−loup Gailly and Mark Adler. Table metadata (column names and attributes) are never compressed and the resulting file remains a structured interchange file. This file can also be processed using the RMU Dump Export command.

In past releases, it was possible that table data, stored in the database with compression enabled, would be many times smaller in the database than when unloaded by RMU. In the database, a simple and fast RLE (run−length encoding) algorithm is used to store rows but this data is fully expanded by RMU Unload. Allowing compression allows the result data file to be more compact using less disk space and permitting faster transmission over communication lines.

Changes to RMU Unload

A new /COMPRESSION qualifier has been added to RMU Unload. The default remains /NOCOMPRESSION. This qualifier accepts the following optional keywords: LZW, ZLIB, LEVEL and EXCLUDE_LIST. The compression algorithms used are ZLIB (the default) or LZW. ZLIB allows further tuning with the LEVEL option that accepts a numeric level between 1 and 9. The default of 6 is usually a good trade off between result file size and the CPU cost of the compression.

It is possible that data in LIST OF BYTE VARYING columns is already in a compressed format (for instance images as JPG data) and therefore need not be compressed by RMU Unload. In fact, compression in such cases might actually cause the output to grow. Therefore, the /COMPRESSION qualifier accepts an option EXCLUDE_LIST which will disable compression for LIST OF BYTE VARYING columns. Specific column names can be listed or, if omitted, all LIST OF BYTE VARYING columns will be excluded from compression.

```
$ rmu/unload/compress=LZW/debug=trace complete_works complete_works
complete_works
Debug = TRACE
Compression = LZW
* Synonyms are not enabled
Unloading Blob columns.
Row_Count = 500
Message buffer: Len: 54524
Message buffer: Sze: 109, Cnt: 500, Use: 31 Flg: 00000000
** compress data: input 2700 output 981 deflate 64%
** compress TEXT_VERSION : input 4454499 output 1892097 deflate 58%
** compress PDF_VERSION : input 274975 output 317560 deflate −15%
%RMU−I−DATRECUNL,   30 data records unloaded.
```

In this example, the column PDF_VERSION contains data that does not compress and so should be excluded on the command line.

```
$ rmu/unload/compress=(LZW,exclude_list:PDF_VERSION)/debug=trace complete_works
complete_works complete_works
Debug = TRACE
Compression = LZW
Exclude_List:
        Exclude column PDF_VERSION
* Synonyms are not enabled
Unloading Blob columns.
Row_Count = 500
Message buffer: Len: 54524
Message buffer: Sze: 109, Cnt: 500, Use: 31 Flg: 00000000
** compress data: input 2700 output 981 deflate 64%
** compress TEXT_VERSION : input 4454499 output 1892097 deflate 58%
%RMU-I-DATRECUNL,   30 data records unloaded.
```

Note

Short rows and short null byte vectors will cause compression to be automatically disabled for the table. However, compression of LIST OF BYTE VARYING columns will still be performed.

The /COMPRESSION qualifier accepts either LZW or ZLIB as the compression method. ZLIB is the default and tends to do the best general compression. However, after testing, the database administrator may decide to use LZW method.

Changes to RMU Load

No new qualifiers are required by RMU Load. The metadata in the interchange file defines the compression algorithm used by RMU Load and indicates which LIST OF BYTE VARYING columns were compressed by RMU Unload.

Changes to RMU Dump Export

A new /OPTIONS qualifier has been added to RMU Dump Export. The default is NOOPTIONS. It accepts the keyword HEADER_SECTION which allows the database administrator to display just the header portion of the interchange file and avoid dumping the data or metadata for every row in the table.

```
$ RMU/DUMP/EXPORT/OPTION=HEADER JOBS.UNL

BEGIN HEADER SECTION - (0)
    NONCORE_TEXT HDR_BRP_ID - (20) :  Load/Unload utility
    CORE_NUMERIC HDR_BRPFILE_VERSION - (1) : 4
    NONCORE_TEXT HDR_DBS_ID - (18) : Oracle Rdb V7.2-10
    NONCORE_TEXT HDR_DB_NAME - (16) : DB$:MF_PERSONNEL
    NONCORE_DATE HDR_DB_LOG_BACKUP_DATE - (8) :  3-JUL-2006 16:52:32.83
    CORE_NUMERIC HDR_DATA_COMPRESSION - (1) : 1
END HEADER SECTION - (0)
```

Here HDR_DATA_COMPRESSION indicates that compression has been used by RMU Unload.

Usage Notes

- Only the user data is compressed. Therefore, additional compression may be applied using various third party compression tools, such as ZIP. It is not the goal of RMU to replace such tools.

- The qualifier RECORD_DEFINITION (or RMS_RECORD_DEF) is not compatible with /COMPRESSION. Note that the TRIM option for DELIMITED format can be used to trim trailing spaces from VARCHAR data.
- The LEVEL keyword may not be used with LZW compression technique. Only one of LZW or ZLIB may be specified for the /COMPRESSION qualifier.

6.1.11 RMU /UNLOAD /AFTER_JOURNAL Commit Information Includes Username

Enhancement 5128647

The Oracle Rdb LogMiner (tm) feature is now able to extract username information. The "C"ommit information record has been extended to include a 12 byte username string. To specify that commit information records are to be extracted to the output stream, specify the COMMIT keyword to the /INCLUDE=ACTION=qualifier.

In DUMP output format, a new field RDB$LM_USERNAME includes the username information. In DELIMITED_TEXT format, a new field is included at the end of the existing record. In TEXT and BINARY format, the username field is added as 12 bytes at the end of the record.

The following example demonstrates using the DUMP output format.

```
$ RMU /UNLOAD /AFTER_JOURNAL SQL$DATABASE AIJ1.AIJBCK –
        /INCLUDE=ACTION=COMMIT –
        /FORMAT=DUMP –
        /TABLE=(NAME=FOO,OUTPUT=FOO.DAT)
$ SEARCH /NOHEAD FOO.DAT RDB$LM_USERNAME
RDB$LM_USERNAME                 : JONES
RDB$LM_USERNAME                 : SMYTHE
```

6.1.12 SHOW DOMAIN and SHOW TABLE Have Better Formatting of DEFAULT Strings

The output from the SHOW DOMAIN and SHOW TABLE command has changed with respect to the DEFAULT values that are strings. In prior versions, the default string was displayed without delimiters which made it hard to read, especially if the default value was all spaces. Additionally, strings from different character sets were not identified.

This release of SQL now displays these strings in quotes and prefixes it with the character set name, unless the character set is the session default.

The following example shows the revised output.

```
SQL> show domain STREET_NAME;
STREET_NAME                     CHAR(40)
 Oracle Rdb default: '>>'
SQL>
SQL> show table (column) PERSON;
Information for table PERSON
```

```
Columns for table PERSON:
Column Name                    Data Type        Domain
-----------                    ---------        ------
LAST_NAME                      CHAR(50)
 Oracle Rdb default: ' '
LATIN_NAME                     VARCHAR(30)
         ISOLATIN1 30 Characters,  30 Octets
 Oracle Rdb default: ISOLATIN1' '

SQL>
```

6.1.13 CALL Statement From Trigger Action Can Now Update Tables

Bug 2421356

In prior releases of Oracle Rdb, the CALL statement could only SELECT data from other tables. With this release of Rdb, the CALL statement may INSERT, DELETE and UPDATE tables as well as CALL other routines. The following restrictions apply to the actions of the routines activated by the CALL statement:

- The table which is the target for the trigger, known as the morphing table, may not be updated (meaning INSERT, DELETE or UPDATE) by any stored procedure or function called within the scope of the trigger activation. Morphing table updates must be done within a trigger definition so that anomalies can be detected and avoided. Attempts to update the morphing tables will result in a runtime error such as the following:

  ```
  %RDB−E−READ_ONLY_REL, relation T was reserved for read access; updates not
  allowed
  −RDMS−E−RTN_ERROR, routine "SET_LENGTH" generated an error during execution
  −RDMS−F−INVTRGACT_STMT, invalid trigger action statement − can not modify
  target table
  ```

  As far as the stored procedure is concerned, the morphing table is a read−only.
- If a stored routine action causes a different trigger to be activated and that then causes the same routine to be called, then an error similar to the following will be raised:

  ```
  %RDB−F−ACTIVE_RTN, routine "CAST_VALUE" is already active
  −RDMS−E−NORECURSION, no recursive routine calls permitted
  ```

Note

A stored routine may only be called from a trigger if it has been analyzed by Oracle Rdb. This step is automatically done by CREATE and ALTER TRIGGER ... ADD statements. If the routine was not recently created in the database (since Oracle Rdb Release 7.0.6), then use the ALTER MODULE ... COMPILE option to recompile any routines.

6.1.14 Using OpenVMS Reserved Memory Registry With Rdb

For Oracle Rdb memory−resident global sections (either row cache global sections or the database root global section), it is possible to utilize the OpenVMS Reserved Memory Registry feature to reserve physical memory. This reserved memory can be useful to allow the use of granularity hint (GH) regions which can further improve performance by using fewer processor translation buffer entries to map a large range of physical memory pages. Use of the reserved memory is optional and any performance gains are application specific.

In order to take advantage of the OpenVMS Reserved Memory Registry feature, global sections must be configured as "SHARED MEMORY IS PROCESS RESIDENT". This can be done with SQL statements "ALTER CACHE ... SHARED MEMORY IS PROCESS RESIDENT" and "ALTER DATABASE ... SHARED MEMORY IS PROCESS RESIDENT".

The name of the global section is required in order to register a global section in the OpenVMS shared memory registry. The "RMU/DUMP/HEADER" command can be used to display the global section names for the database root global section and the row cache global sections. This command also displays the size of the global sections in megabytes rounded up to the next whole megabyte.

For example, information about a row cache global section in the output from the RMU/DUMP/HEADER command might include the following:

```
Shared Memory...
  − Shared memory will be mapped resident
  − Global Section Name is "RDM72R$1$DGA2031064003D000000000005"
  − Shared memory section requirement is 77,070,336 bytes (74MB)
```

Information about the database global section in the output from the RMU/DUMP/HEADER command might include the following:

```
Derived Data...
  − Global section size
      With global buffers disabled is 2,047,042 bytes (2MB)
      With global buffers enabled is 33,860,114 bytes (33MB)
        .
        .
        .
  − Global Section Name is "RDM72N$1$DGA2031064003D000000000000"
```

From these examples, the row cache section size would be 74 megabytes and the database global section size (with global buffers enabled) would be 33 megabytes.

To reserve the memory, use the SYSMAN utility RESERVED_MEMORY ADD command and then run AUTOGEN as in the following examples:

```
$ RUN SYS$SYSTEM:SYSMAN
SYSMAN> RESERVED_MEMORY ADD RDM72N$1$DGA2031064003D000000000000 −
        /ALLOCATE /SIZE=33
SYSMAN> RESERVED_MEMORY ADD RDM72R$1$DGA2031064003D000000000005 −
        /ALLOCATE /SIZE=74
SYSMAN> EXIT
$ @SYS$UPDATE:AUTOGEN ...
```

The OpenVMS system must then be shutdown and restarted for the memory reservations to be in effect.

After rebooting and reopening databases, the SHOW MEMORY /RESERVED command can be used to see that the reserved memory is in use. For example:

```
$ SHOW MEMORY/RESERVED
Memory Reservations (pages):       Group  Reserved   In Use       Type
 RDM72R$1$DGA408451A6A000000000002
                                   SYSGBL          2        2  Page Table
 RDM72R$1$DGA408451A6A000000000002
                                   SYSGBL       1536     1353   Allocated
 Total (12.01 MBytes reserved)                  1538     1355
```

Database Root File Specific

Changes to the size of the database or row cache global sections will require that the memory reservation size be updated (either by removing and re−adding or modifying the existing reservation). Further, because the device and file identification of the database root file are encoded in the global section names, any operation (such as restoring or moving) that changes either the file identification or the device identification of the root file will result in the global section names changing.

If the reserved memory is specified with a size smaller than the actual size of the global section, the section may fail to be created when the database is opened or accessed with a message similar to "SYSTEM−F−INSFLPGS, insufficient Fluid Pages available".

For further information, review the OpenVMS documentation set including "HP OpenVMS System Manager's Manual, Volume 2: Tuning, Monitoring, and Complex Systems", "HP OpenVMS Version 8.2−1 for Integrity Servers New Features and Release Notes", and "HP OpenVMS System Services Reference Manual".

6.1.15 Server Output File Names As Database Attributes

Previously, logical names could be used to control various server output or log file names and locations. In many cases, these logical names would have to be defined system−wide and thus could effect the servers of multiple databases.

This situation has been improved. The output or log file names for a number of database server processes are now also controlled by optional database attributes.

The "RMU /SET SERVER /OUTPUT=filespec servertype" command can be used to specify the default output file specification for several of the database server processes. Existing logical names are still valid and supported and will override the database attribute if defined. If the output file specification is empty, the entry is disabled.

Valid values for the "servertype" parameter and the matching logical name are:

Table 6−2 Server Types and Logical Names

Server	Servertype	Logical Name

AIJ Backup Server	ABS	RDM$BIND_BIND_ABS_OUTPUT_FILE
AIJ Log Server	ALS	RDM$BIND_BIND_ALS_OUTPUT_FILE
AIJ Log Roll−Forward Server	LRS	RDM$BIND_LRS_OUTPUT_FILE
AIJ Log Catch−Up Server	LCS	RDM$BIND_LCS_OUTPUT_FILE
Database Recovery Server	DBR	RDM$BIND_DBR_LOG_FILE
Row Cache Server	RCS	RDM$BIND_RCS_LOG_FILE

The /LOG qualifier can be used to display a log message at the completion of the RMU /SET operation.

Examples of using the "RMU /SET SERVER /OUTPUT=filespec servertype" command follow.

```
$ RMU /SET SERVER RCS /OUTPUT=RCS_PID.LOG /LOG DUA0:[DB]MYDB.RDB
$ RMU /SET SERVER ALS /OUTPUT=ALS$LOGS:ALS_DB1.LOG DUA0:[DB1]MFP.RDB
$ RMU /SET SERVER LRS /OUTPUT="" DUA0:[ZDB]ZDB.RDB
$ RMU /SET SERVER DBR /OUTPUT=DBR$LOGS:DBR.LOG DUA0:[ADB]ADB.RDB
```

6.1.16 New REBLDSPAM Informational Message Added to RMU/VERIFY

An informational "REBLDSPAM" message has been added to RMU/VERIFY which is output when the database Area Inventory (AIP) pages, which contain information about the database logical areas where table data and indexes are stored, are verified. This message is output if a flag is set for a logical area that indicates that the Space Management (SPAM) pages for the logical area should be updated to reflect changes to space thresholds or record lengths that may have been made by an SQL ALTER TABLE or other command. This message is INFORMATIONAL since updating the SPAM pages is not essential for the integrity or functionality of the database but can improve performance. The SPAM pages can be updated using the RMU/REPAIR command.

The following shows an example of this message which includes the name and id number of the logical area where the SPAM pages should be rebuilt to improve performance.

```
$RMU/VERIFY/ALL MF_PERSONNEL
%RMU−I−REBLDSPAM, Space management (SPAM) pages should be rebuilt for
                  logical area DEPARTMENTS_INDEX,
                  logical area id 74
%RMU−I−REBLDSPAM, Space management (SPAM) pages should be rebuilt for
                  logical area DEPARTMENTS,
                  logical area id 75
```

6.1.17 Increased Date/Time String Display Precision

For several values where there is enough space on the display, the RMU SHOW STATISTICS Utility now displays time/date stamps with precisions greater than 0.01 second units. In several cases (stall displays, for example), the screen display width must be 100 or more columns in order to display the full date/time with seven fractional digits.

For example, the "short" time and/or date format displays include only two fractional digits:

- 16:23:16.17
- 13−NOV−2006 16:23:16.17

While the "long" time and/or date format displays include seven fractional digits:

- 16:23:16.1776975
- 13−NOV−2006 16:23:16.1776975

6.1.18 Enhanced System Table Lookup in Multischema Databases

In prior releases of Oracle Rdb, applications that attached to a multischema database had to explicitly query the Rdb system tables using the catalog and schema name RDB$CATALOG.RDB$SCHEMA. Otherwise, a SET SCHEMA statement by the application might cause these system queries to fail. This was particularly a problem with interfaces such as SQL/Services and the Oracle ODBC Driver for Rdb.

With this release, Oracle Rdb will first try to locate the table in the default schema as established by the SET CATALOG, SET SCHEMA or ATTACH statements. If the lookup fails, Rdb will try RDB$CATALOG.RDB$SCHEMA. This lookup will apply to tables, sequences, functions and procedures for both system and user defined objects.

The following example shows the successful query with this new functionality.

```
SQL> attach 'filename db$:msdb';
SQL>
SQL> set schema 'west';
SQL>
SQL> select rdb$relation_name
cont> from rdb$relations
cont> where rdb$relation_name like 'JOB%';
RDB$RELATION_NAME
JOBS
JOB_HISTORY
2 rows selected
SQL>
```

The same query in an older version would fail.

```
SQL> attach 'filename db$:msdb';
SQL>
SQL> set schema 'west';
SQL>
SQL> select rdb$relation_name
cont> from rdb$relations
cont> where rdb$relation_name like 'JOB%';
%SQL−F−RELNOTDEF, Table RDB$RELATIONS is not defined in database or schema
SQL>
```

This problem has been corrected in Oracle Rdb Release 7.2.1.

6.1.19 New SET FLAGS Option: REBUILD_SPAM_PAGES

A new flag, REBUILD_SPAM_PAGES, has been added for use in conjunction with the DDL commands ALTER TABLE, ALTER STORAGE MAP, and ALTER INDEX.

When changing the row length or THRESHOLDS clause for a table or index, the corresponding SPAM pages for the logical area may require rebuilding. By default, these DDL commands update the AIP and set a flag to indicate that the SPAM pages should be rebuilt. However, this new flag may be set prior to executing a COMMIT for the transaction and the rebuild will take place within this transaction.

Use SET FLAGS 'NOREBUILD_SPAM_PAGES' to negate this flag.

The following example shows a simple change to the EMPLOYEES table (mapped in this example to a set of UNIFORM areas). The flag STOMAP_STATS is used to enable more trace information from the ALTER and COMMIT statements.

```
SQL> set transaction read write;
SQL>
SQL> set flags 'stomap_stats';
SQL>
SQL> alter table EMPLOYEES
cont>     add column MANAGERS_COMMENTS varchar(300);
~As: reads: async 0 synch 94, writes: async 18 synch 1
SQL>
SQL> alter storage map EMPLOYEES_MAP
cont>     store
cont>         using (EMPLOYEE_ID)
cont>             in EMPIDS_LOW
cont>           (thresholds (34,76,90))
cont>                with limit of ('00200')
cont>             in EMPIDS_MID
cont>           (thresholds (34,76,90))
cont>                with limit of ('00400')
cont>             otherwise in EMPIDS_OVER
cont>           (thresholds (34,76,90));
~As locking table "EMPLOYEES" (PR -> PU)
~As: removing superseded routine EMPLOYEES_MAP
~As: creating storage mapping routine EMPLOYEES_MAP (columns=1)
~As: reads: async 0 synch 117, writes: async 56 synch 0
SQL>
SQL> set flags 'rebuild_spam_pages';
SQL>
SQL> commit;
%RDMS-I-LOGMODVAL,     modified record length to 423
%RDMS-I-LOGMODVAL,     modified space management thresholds to (34%, 76%, 90%)
%RDMS-I-LOGMODVAL,     modified record length to 423
%RDMS-I-LOGMODVAL,     modified space management thresholds to (34%, 76%, 90%)
%RDMS-I-LOGMODVAL,     modified record length to 423
%RDMS-I-LOGMODVAL,     modified space management thresholds to (34%, 76%, 90%)
SQL>
```

The message LOGMODVAL will appear for each logical area in the storage map, one per partition.

This rebuild action only applies to UNIFORM storage areas and may incur significant I/O as SPAM pages and data pages are read to allow the SPAM page to be rebuilt.

6.1.20 RMU/BACKUP /NORECORD New Qualifier

A new qualifier has been added which avoids the modification of the database with recent backup information. Hence the database appears like it had not been backed up at this time.

The main purpose of this qualifier is to allow a backup of a hot standby database without modifying the database files.

Example using the /NORECORD qualifier:

```
$ RMU /BACKUP /NORECORD FOO BCK
```

6.1.21 Improved Management of the AIP (Area Inventory Page) Data by SQL Commands

Bugs 4007253, 3840715, 3019205 and 4861228

This release of Oracle Rdb changes the behavior of several DDL (data definition language) commands so that they now maintain information in the AIP (area inventory pages).

- Changed behavior for ALTER TABLE
  In prior releases of Oracle Rdb, the record length in the AIP (area inventory pages) was set when the table was created. Subsequent ALTER TABLE statements that added new columns, changed column length or data types, or dropped columns would not update this length.
  If the record length on the AIP became too out−of−date, then INSERT performance could be affected when a target page had enough room for an original row but not for the current row size. Commands such as RMU/REPAIR/INITIALIZE=LAREA_PARAMETERS/SPAM could be used to reset this length and rebuild SPAM (space management) pages that referenced the table. However, the database administrator had to calculate the revised length and be aware that the SPAM pages would need to be rebuilt for best performance.
  With this release of Oracle Rdb, the ALTER TABLE statement will track changes in the length of the table row. All such changes during a transaction are considered and, if there is an overall change in length from that currently saved in the AIP, then Rdb will update the AIP page and flag the logical area (or logical areas) so that at a convenient time the SPAM pages can be rebuilt.
  If there is no net row length change made during the transaction then no attempt is made to update the AIP or SPAM pages. Updates to the AIP are immediate since there is no SNAPSHOT support for the AIP. Therefore, these actions to update the AIP are deferred until COMMIT time.

Note

The record length as recorded in the AIP can change when:

- ◊ ***A new column is added to the table (ALTER TABLE ... ADD COLUMN)***
- ◊ ***An existing column is dropped from a table (ALTER TABLE ... DROP COLUMN)***
- ◊ ***A data type of a column is changed to one of a different size***
- ◊ ***The data type of the column remains the same (CHAR and VARCHAR) but the length is changed***

◊ ***The RDB$FIELD_ID field increases such that the NBV (null bit vector) for the row must be expanded.***

- Changed behavior for RENAME TABLE and RENAME INDEX
  In prior releases of Oracle Rdb, the name of the logical area was not changed when a table was renamed. Apart from being confusing when trying to match the logical area names with the table names, it also caused some RMU utilities to compute incorrect values because they assumed the table name was matched by the logical area name.
  With this release of Oracle Rdb, the ALTER TABLE ... RENAME TO and RENAME TABLE statement will also revise the name of the logical area.
- Changed behavior for TRUNCATE TABLE
  Truncate table will implicitly update the AIP record length for the table. There is no need to rebuild the SPAM pages because TRUNCATE removes all rows from the table which implies there will be no SPAM references to any rows. Subsequent inserts will use the new, revised record length when searching for free space.
- Changed behavior for ALTER STORAGE MAP and ALTER INDEX
  In prior releases of Oracle Rdb, the THRESHOLDS ARE clause could not be applied to an existing partition. Any such attempts caused an error similar to the following:

```
SQL> alter storage map sample_table_map
cont>     store using (a)
cont>           in U_EMPIDS_LOW (thresholds are (31,41,81)) with limit of (10)
cont> ;
%RDB-E-NO_META_UPDATE, metadata update failed
-RDMS-E-THRESHAREEXI, illegal thresholds usage - area U_EMPIDS_LOW exists,
and cannot have THRESHOLDS respecified
```

  With this release of Oracle Rdb, the ALTER STORAGE MAP and ALTER INDEX statements will now allow this change to succeed. The new thresholds will be applied and the logical area (or logical areas) will be flagged so that at a convenient time the SPAM pages can be rebuilt.

These actions to update the AIP for length, thresholds and name are deferred until COMMIT time. Therefore, it is possible that the COMMIT following TRUNCATE TABLE may perform additional I/O if the nominal record length in the AIP differs from the actual record length of the current row version.

If there is no net row length change made during the transaction and no change of name or threshold, then Rdb will not attempt to update the AIP pages.

These problems have been corrected in Oracle Rdb Release 7.2.1.

6.1.22 New RMU Show AIP Command Added

This release of Oracle Rdb adds a new command that displays the contents of the AIP (Area Inventory Pages) structure. The AIP structure provides a mapping for logical areas to physical areas as well as describing each of those logical areas. Information such as the logical area name, length of the stored record, storage thresholds and other information can now be displayed using this simple command interface. In prior versions, the RMU Dump Larea=RDB$AIP command was the only RMU command that displayed this information.

Format

RMU/SHOW AIP rootfile [larea−name] [/LAREA=(n [,...])] [/OPTION=REBUILD_SPAMS] [/OUTPUT=output−filename] [/TYPE=type−name]

Description

The RMU Show AIP command allows the database administrator to display details of selected logical areas or all logical areas in the database.

Command Parameters

- root−file−spec
  The file specification for the database root file to be processed. The default file extension is .rdb.
- larea−name
  An optional parameter that allows the logical areas to be selected by name. Only those AIP entries are displayed. This parameter is optional and will default to all logical areas being displayed.
  Any partitioned index or table will create multiple logical areas all sharing the same name. This string may contain standard OpenVMS file card characters (% and *) so that different names can be matched. Therefore, it is possible for many logical areas to match this name. A list of logical area names cannot be specified.
  The value of ***larea−name*** may be delimited so that mixed case characters, punctuation and various character sets can be used.

Command Qualifiers

- Larea
  Specifies a list of logical area identifiers. The LAREA qualifier and larea−name parameter are mutually exclusive. The default if neither the LAREA qualifier nor the larea−name parameter is specified is to display all AIP entries.
- Output [= outout−filename]
  This qualifier is used to capture the output in a named file. If used, a standard RMU header is added to identify the command and database being processed. If omitted, the output is written to SYS$OUTPUT and no header is displayed.
- Option = REBUILD_SPAMS
  Display only those logical areas which have the REBUILD_SPAMS flag set.
- Type = type−name
  Legal values for type−name are TABLE, SORTED_INDEX, HASH_INDEX, LARGE_OBJECT, and SYSTEM_RECORD.
  This qualifier is used in conjunction with ***larea−name*** to select a subset of the AIP entries that may match a name. For instance, it is legal in Rdb to create a table and an index with the name EMPLOYEES. So using EMPLOYEES/TYPE=TABLE will make the selection unambiguous. It also allows simpler wildcarding. Commands using *EMPLOYEE*/TYPE=TABLE will process only those tables that match and not the associated index logical areas.

Usage Notes

- The database administrator requires RMU$DUMP privilege as this command is closely related to the RMU DUMP LAREA=RDB$AIP command.
- Only AIP entries that are in use are displayed. In contrast, the RMU Dump command also displays deleted and unused AIP entries.

Examples

This example uses the name of a known database table to display details for this single logical area.

Example 6−1 Displaying the AIP entry for the JOBS table

```
$ RMU/SHOW AIP SQL$DATABASE JOBS

Logical area name JOBS
Type: TABLE
Logical area 85 in mixed physical area 7
Physical area name JOBS
Record length 41
AIP page number: 151
ABM page number: 0
Snapshot Enabled TSN: 64
```

The wildcard string "*EMPLOYEE* matches both indices and table logical areas so here we use /TYPE to limit the display to just table logical areas. The table EMPLOYEES in the MF_PERSONNEL database is partitioned across three storage areas and hence there exists three logical areas.

Example 6−2 Using wildcards and /TYPE qualifier

```
$ RMU/SHOW AIP SQL$DATABASE *EMPLOYEE*/TYPE=TABLE

Logical area name EMPLOYEES
Type: TABLE
Logical area 80 in mixed physical area 3
Physical area name EMPIDS_LOW
Record length 126
AIP page number: 150
ABM page number: 0
Snapshot Enabled TSN: 4800

Logical area name EMPLOYEES
Type: TABLE
Logical area 81 in mixed physical area 4
Physical area name EMPIDS_MID
Record length 126
AIP page number: 151
ABM page number: 0
Snapshot Enabled TSN: 1504

Logical area name EMPLOYEES
Type: TABLE
Logical area 82 in mixed physical area 5
Physical area name EMPIDS_OVER
Record length 126
AIP page number: 151
ABM page number: 0
Snapshot Enabled TSN: 1504
```

This example shows the REBUILD_SPAMS option used to locate logical areas that require SPAM rebuilds. This may occur because the stored row length changed size or THRESHOLDS were modified for the index or storage map.

Example 6−3 Locating AIP entries that need rebuilding

```
$ RMU/SHOW AIP/OPTION=REBUILD_SPAMS
_Root: SQL$DATABASE
_Logical area name:

Logical area name ACCOUNT_AUDIT
Type: TABLE
Logical area 86 in uniform physical area 1
Physical area name RDB$SYSTEM
Record length 12
Thresholds are (10, 100, 100)
Flags:
    SPAM pages should be rebuilt
AIP page number: 151
ABM page number: 1004
Snapshot Enabled TSN: 5824

Logical area name DEPARTMENTS_INDEX
Type: SORTED INDEX
Logical area 94 in uniform physical area 10
Physical area name DEPARTMENT_INFO
Record length 430
Thresholds are (30, 65, 72)
Flags:
    SPAM pages should be rebuilt
AIP page number: 151
ABM page number: 2
Snapshot Enabled TSN: 7585
```

6.1.23 New RMU Set AIP Command Added

This release of Oracle Rdb adds a new command that modifies the contents of the AIP (Area Inventory Pages) structure. The AIP structure provides a mapping for logical areas to physical areas as well describing each of those logical areas. Information such as the logical area name, length of the stored record, and storage thresholds can now be modified using this simple command interface. In prior versions, the RMU Repair Initialize=Larea_Parameters command was the only RMU command that allowed updates to this information.

Format

RMU/SET AIP root−file−spec larea−name [/LAREA=(n [, ...])] [/LENGTH[=n]] [/LOG] [/REBUILD_SPAMS] [/RENAME_TO=new−name] [/THRESHOLD=(p,q,r)]

Description

This RMU command is used to modify some attributes of an existing logical area. It cannot be used to add or delete a logical area. This command can be used to correct the record length, thresholds and name of a logical area described by an AIP entry. It can also be used to rebuild the SPAM pages for a logical area stored in UNIFORM page format areas so that threshold settings for a page correctly reflect the definition of the table.

See also the RMU Repair Spam command for information on rebuilding SPAM pages for MIXED areas.

Command Parameters

- root−file−spec
  The file specification for the database root file to be processed. The default file extension is .rdb.
- larea−name
  An optional parameter that allows the logical areas to be selected by name. Only those AIP entries are processed.
  Any partitioned index or table will create multiple logical areas all sharing the same name. This string may contain standard OpenVMS file card characters (% and *) so that different names can be matched. Therefore, it is possible for many logical areas to match this name. A list of logical area names cannot be specified.
  The value of ***larea−name*** may be delimited so that mixed case characters, punctuation and various character sets can be used.

Command Qualifiers

- Larea = (n1 [, n2 ...])
  Specifies a list of logical area identifiers. The LAREA qualifier and larea−name parameter are mutually exclusive.
- Length [= value]
  Sets the length of the logical area. If no value is provided on the RMU Set AIP command, then Oracle Rdb will find the matching table and calculate a revised AIP nominal record length and apply it to the AIP.
- Log
  Logs the names and identifiers of logical areas modified by this command.
- Rebuild_Spams
  Locate each logical area with the "rebuild−spam" flag set and rebuild the SPAM pages.
- Rename_To = new−name
  Used to change the logical area name. This qualifier should be used with caution as some RMU commands assume a strict mapping between table/index names and names of the logical area. This command can be used to repair names that were created in older versions of Oracle Rdb where the rename table command did not propagate the change to the AIP. The value of new−name may be delimited so that mixed case, punctuation and various character sets can be used.
- Threshold = (t1 [,t2 [, t3]])
  Changes the threshold on all logical areas specified using the Larea qualifier or the larea−name parameter. RMU accepts THRESHOLD=(0,0,0) as a valid setting to disable logical area thresholds. Values must be in the range 0 through 100. Any missing values default to 100.

Usage Notes

- The database administrator requires RMU$ALTER privilege to run the command and the Rdb server also requires SELECT and ALTER privilege on the database.
- This command supersedes the RMU Repair Initialize=Larea_Parameters command that can also change the Thresholds and Length for a logical area. This command can be executed online, where as the RMU Repair command must be run offline.
- Wildcard names are not permitted with the following qualifiers to prevent accidental propagation of values to the wrong database objects.
  - ♦ LENGTH qualifier with a value is specified,
  - ♦ RENAME_TO qualifier,
  - ♦ and THRESHOLDS qualifier.
- RMU Set AIP may be used on a master database configured for HOT STANDBY. All AIP changes and SPAM rebuild actions are written to the after image journal and will be applied to the standby database. This command cannot be applied to a STANDBY database.

- THRESHOLDS for MIXED format areas are physical area attributes and are not supported at the logical area (aka AIP) level. Therefore, THRESHOLDS can not be applied to MIXED areas and specifying logical areas will cause an exception to be raised.
- The REBUILD_SPAMS qualifier is only applied to logical areas stored in UNIFORM page format storage areas.
- This command will implicitly commit any changes with no opportunity to undo them using rollback. Access to the functionality is controlled by privileges at the RMU and Rdb database level. We suggest that RMU Show AIP be used prior to any change so that you can compare the results and repeat the RMU Set AIP command with corrections if necessary.
  Some wildcard operations are restricted to prevent accidental damage to the database. For instance, a wildcard matching many objects will be rejected if more than one type of object is being changed. If a wildcard selects both table and index types, then this command will be rejected.
- This command is an online command. Each logical area will be processed within a single transaction and interact with other online users.
- When the AIP entry is changed online, any existing users of the table or index will start to use the new values if the logical areas are reloaded.
- Various SQL alter commands will register changes for the AIP and these are applied at COMMIT time. RMU Verify and RMU Show AIP Option=REBUILD_SPAMS will report any logical areas that require SPAM rebuilding. The database administrator can also examine the output from the RMU Dump Larea=RDB$AIP command.
- How long can the SPAM rebuild be delayed? The fullness of some page will have been calculated using the old AIP length or THRESHOLD values. Therefore, it might appear that a page is full when in fact the revised length will fit on the page, or the page may appear to have sufficient free space to store a row but once accessed the space is not available. By rebuilding SPAM pages, you may reduce I/O during insert operations. However, delaying the rebuild to a convenient time will not affect the integrity of the database.
- The amount of I/O required for Rebuild_Spams depends upon the number of pages allocated to the table or index involved. Assuming just one logical area is selected, then Oracle Rdb will read the ABM (Area Bitmap) to locate all SPAM pages in that area that reference this logical area. Rdb will then read each page in the SPAM interval for that SPAM page and recalculate the fullness based on the rows stored on each page.

Examples

RMU will call Rdb for each logical area that requires rebuilding.

Example 6−4 Rebuilding SPAM pages for logical areas

```
$ RMU/SET AIP/REBUILD_SPAMS MF_PERSONNEL
%RMU-I-AIPSELMOD, Logical area id 86, name ACCOUNT_AUDIT selected for
modification
%RMU-I-AIPSELMOD, Logical area id 94, name DEPARTMENTS_INDEX selected for
modification
```

RMU will request that the EMPLOYEES table length be updated in the AIP. Oracle Rdb will use the latest table layout to calculate the length in the AIP and write this back to the AIP. The EMPLOYEES table is partitioned across three storage areas and therefore the Log qualifier shows these three logical areas being updated.

Example 6−5 Updating the length in the AIP for a table

```
$ RMU/SET AIP MF_PERSONNEL EMPLOYEES/LENGTH/LOG
%RMU-I-AIPSELMOD, Logical area id 80, name EMPLOYEES selected for modification
%RMU-I-AIPSELMOD, Logical area id 81, name EMPLOYEES selected for modification
%RMU-I-AIPSELMOD, Logical area id 82, name EMPLOYEES selected for modification
```

RMU will request that the EMPLOYEES table length be updated in the AIP and then the SPAM pages will be rebuilt. This is an ONLINE operation. Note: there is an implied relationship between the logical area name and the name of the object. This example assumes that the EMPLOYEES object is mapped to a UNIFORM page format area.

Example 6-6 Updating the length for a table and rebuilding SPAM pages

```
$ RMU/SET AIP MF_PERSONNEL EMPLOYEES/LENGTH/REBUILD_SPAMS
```

When thresholds for an index are modified, they will not be effective until the SPAM pages are updated (rebuilt) to use these new values. The following example shows the index maintenance performed by SQL. The SET FLAGS command is used to display information about the change. Note that the change is applied at COMMIT time and that the SPAM rebuild is deferred until a later time. RMU Set AIP is then used to rebuild the SPAM pages.

Example 6-7 Updating the thresholds for a SORTED index

```
$ SQL$
SQL> set flags 'index_stats';
SQL> alter index candidates_sorted store in rdb$system (thresholds are (32,56,
77));
~Ai alter index "CANDIDATES_SORTED" (hashed=0, ordered=0)
~Ai larea length is 215
~As locking table "CANDIDATES" (PR -> PU)
~Ai: reads: async 0 synch 58, writes: async 8 synch 0
SQL> commit;
%RDMS-I-LOGMODVAL,     modified space management thresholds to (32%, 56%, 77%)
%RDMS-W-REBUILDSPAMS, SPAM pages should be rebuilt for logical area CANDIDATES_
SORTED
$
$ RMU/SET AIP MF_PERSONNEL CANDIDATES_SORTED/REBUILD_SPAMS/LOG
%RMU-I-AIPSELMOD, Logical area id 74, name CANDIDATES_SORTED selected for
modification
```

Chapter 7
Enhancements And Changes Provided in Oracle Rdb Release 7.2.0.2

7.1 Enhancements And Changes Provided in Oracle Rdb Release 7.2.0.2

7.1.1 Enhancements to Concurrent DDL Statements

Bug 4761143

In prior versions of Oracle Rdb, attempts to run several ALTER TABLE ... ADD CONSTRAINT commands in different sessions in parallel would either stall waiting for another transaction to finish or fail with a deadlock as shown in the following example.

```
SQL> alter table ORDER_LINES
cont>   add constraint ORDER_LINES_FK
cont>      foreign key (order_number) references orders (order_number)
cont>      not deferrable
cont> ;
%RDB-E-DEADLOCK, request failed due to resource deadlock
-RDB-E-NO_META_UPDATE, metadata update failed
-RDMS-F-DEADLOCK, deadlock on client '........ORDE'
4544524F0000001F0000000400000055
```

This behavior occurs because of Rdb's locking of the target tables to ensure consistent data in all tables. For example, for the constraint in the example to validate, there must not be another transaction deleting rows from the ORDER_LINES table. Rdb ensures this by locking the metadata for each table referenced in a constraint definition.

To provide better concurrency, this release of Oracle Rdb now allows the following statements to be used when the target table is reserved in DATA DEFINITION mode.

- ALTER TABLE can now reference the table reserved for DATA DEFINITION MODE
  The following clauses are supported:
  - ADD CONSTRAINT
  - ALTER COLUMN ... CONSTRAINT
  - ENABLE CONSTRAINT, ENABLE PRIMARY KEY, and ENABLE UNIQUE (...)
  - DROP CONSTRAINT
  - ALTER COLUMN ... NULL
  - DISABLE CONSTRAINT, DISABLE PRIMARY KEY and DISABLE UNIQUE
  - DISABLE UNIQUE (...)

  The ADD and ENABLE CONSTRAINT are best suited to concurrent execution as they may require I/O to validate the constraint.
- ALTER INDEX can now be used to build all or part of the index on a table reserved for DATA DEFINITION mode.
  The following clauses are supported:
  - BUILD PARTITION and BUILD ALL PARTITIONS
  - REBUILD PARTITION and REBUILD ALL PARTITIONS
  - TRUNCATE PARTITION and TRUNCATE ALL PARTITIONS
  - COMMENT IS clause

The BUILD and REBUILD PARTITION operators are best suited to concurrent execution as they may require I/O to construct the new index partition.

- ALTER VIEW and CREATE VIEW may now reference a table reserved for DATA DEFINITION mode.
- COMMENT ON TABLE can now reference a table reserved for DATA DEFINITION mode.
- The statement DROP CONSTRAINT can now reference a constraint on a table reserved for DATA DEFINITION mode.

Note

In prior releases, only the CREATE INDEX statement was permitted within a transaction that reserved a table in DATA DEFINITION mode.

Most ALTER TABLE clauses are now supported for tables reserved for SHARED DATA DEFINITION. The exceptions are those clauses that change the structure of the table: ADD COLUMN, DROP COLUMN and ALTER COLUMN which changes the data type.

This problem has been corrected in Oracle Rdb Release 7.2.0.2.

7.1.2 RMU Load and Unload Now Support Table and View Synonyms

Bug 4018104

This release of Oracle Rdb, 7.2.0.2, adds support for table and view synonyms for the RMU Load and RMU Unload commands. In prior releases of Rdb, the synonym name was not understood by RMU and resulted in an error, as in the following example.

```
$ SQL$
SQL> show tables
User tables in database with filename db$:personnel
     CANDIDATES
     COLLEGES
     CURRENT_INFO                    A view.
     CURRENT_JOB                     A view.
     CURRENT_SALARY                  A view.
     DEGREES
     DEPARTMENTS
     EMPLOYEES
     JOBS
     JOB_HISTORY
     RESUMES
     SALARY_HISTORY
     WORK_STATUS
     EMPS                            A synonym for table EMPLOYEES
SQL>exit
$ rmu/unload db$:personnel emps emps
%RMU-E-OUTFILDEL, Fatal error, output file deleted
-RMU-F-RELNOTFND, Relation (EMPS) not found
```

These tools now translate the synonym to the base object and process the data as though the base table had been named. This implies that the unload interchange files (.UNL) or record definition file (.RRD) that

contain the table metadata will name the base table or view and not use the synonym name. Therefore, if the metadata is used against a different database you may need to use the /MATCH_NAME qualifier to override this name during RMU Load.

Chapter 8
Enhancements And Changes Provided in Oracle Rdb Release 7.2.0.1

8.1 Enhancements And Changes Provided in Oracle Rdb Release 7.2.0.1

8.1.1 Reduced CPU Usage and Improved Performance

Several performance enhancements have been implemented in this release of Oracle Rdb. Most of these changes are either specific to applications running on I64 systems or will have a greater effect on I64 systems. These enhancements include:

- Streamlined code path of interpreted instructions
- Reduction in use of queue related PAL code and/or PAL system services
- Reduced locking for exclusive database access
- Reduced alignment faults

8.1.2 Enhancements to the ALTER TABLE ... ALTER COLUMN Clause

Bugs 2170476 and 4874525

The ALTER COLUMN clause has been enhanced with this release of Oracle Rdb and now allows columns to be altered to and from COMPUTED BY, AUTOMATIC and IDENTITY special columns.

- ALTER COLUMN may now change an AUTOMATIC column to a normal updateable base column even if there exists constraints and indices, as long as the data types are the same. In prior releases, the presence of a database wide collating sequence prevented this action.
- A non−computed base column can now be altered to be an AUTOMATIC column. The old data is retained and the column is made read−only.
- A non−computed base column can be now be altered to be a COMPUTED BY column. The old data will not be accessible (a warning is issued for interactive SQL) and references to that column will evaluate the COMPUTED BY expression. If indices or constraints reference this column, then the ALTER TABLE statement will fail.
  Note that altering the column back to a base, or automatic, column will allow older versions of the row data to be visible (any rows inserted while the column was a COMPUTED BY column will return NULL).
- The IDENTITY syntax is now supported by ALTER TABLE ... ALTER COLUMN clause.
  If the table has no existing IDENTITY column, a new sequence for the table will be created. Care must be taken to ensure that the IDENTITY will not generate existing values for the column as this would cause INSERT to fail. Use the parameters on IDENTITY to specify an appropriate START WITH value, or modify the sequence using ALTER SEQUENCE.
  If the table has an existing IDENTITY column then an error is raised.
- In some prior versions, a computed column (AUTOMATIC, IDENTITY) could be converted to a non−computed column. However, the dependency rows were never erased from the RDB$INTERRELATIONS table. This is now handled correctly. This is not a serious problem but it may cause unexpected errors when future ALTER and DROP statements are used for those referenced objects.

Please contact Oracle Support for assistance if this problem arises.

- If an IDENTITY column is converted to a base column, a COMPUTED BY column, or AUTOMATIC column, then the special sequence is automatically dropped.
- If a column has a DEFAULT (base column or AUTOMATIC UPDATE AS column) and it is converted to a COMPUTED BY, AUTOMATIC AS or an AUTOMATIC INSERT AS column, then the DEFAULT value is removed (as these types of columns are incompatible with DEFAULT).

8.1.3 /TRANSPORT Added to RMU/REPLICATE AFTER

Bug 4109344

The /TRANSPORT qualifier has been added to the RMU/REPLICATE AFTER START and CONFIGURE commands. This new qualifier allows the network transport to be specified. The valid values are "DECNET" and "TCPIP". The specified network transport is saved in the database.

In previous releases, to use TCP/IP as the network transport for Hot Standby, the system−wide logical "RDM$BIND_HOT_NETWORK_TRANSPORT" had to be defined.

For example:

```
$ RMU/REPLICATE AFTER CONFIGURE /TRANSPORT=TCPIP /STANDBY=
REMNOD::DEV:[DIR]STANDBY_DB M_TESTDB
```

8.1.4 New SHOW STATISTICS Command for Interactive SQL

This release of Oracle Rdb adds a SHOW STATISTICS command to Interactive SQL. This command displays some simple process statistics for the current process and is used primarily to compare resource usage and elapsed time for different queries.

The following example shows the output after performing a typical query.

```
SQL> select count (*)
cont>  from employees natural full outer join job_history;

         274
1 row selected
SQL> show statistics;

                  process statistics at  5−MAR−2006 05:57:48.28
         elapsed time =   0 00:00:00.16              CPU time =   0 00:00:00.05
     page fault count = 430              pages in working set = 22768
   buffered I/O count = 26                   direct I/O count = 83
      open file count = 12               file quota remaining = 7988
           locks held = 138                   locks remaining = 16776821
      CPU utilization = 31.2%             AST quota remaining = 995
SQL>
```

The statistics are reset after each execution of SHOW STATISTICS.

Chapter 9
Enhancements And Changes Provided in Oracle Rdb Release 7.2

9.1 Enhancements And Changes Provided in Oracle Rdb Release 7.2

9.1.1 Default Floating Point Format

The Intel Itanium architecture has a 64−bit virtual address model and basic system functions similar to the Alpha architecture. However, there are some implementation differences between the two platforms that might affect user−written applications.

One of the differences is the availability of hardware−supported floating−point formats. The Intel Itanium architecture implements floating−point arithmetic in hardware using the IEEE floating−point formats, including IEEE single and IEEE double. The Alpha architecture supports both IEEE and VAX floating−point formats in hardware, and OpenVMS compilers generate code using the VAX formats by default, with options (on Alpha) to use IEEE formats. Irrespective of whether it was originally written for VAX or Alpha, an OpenVMS application that uses the default VAX floating−point formats needs to produce equivalent behavior on the Intel Itanium architecture using its native IEEE formats.

- On OpenVMS VAX and OpenVMS Alpha, VAX float is the default. VAX format data is assumed and VAX floating instructions are used.
- On OpenVMS Alpha, you can specify the compiler option /FLOAT=IEEE. In this case, IEEE format data is assumed and IEEE floating instructions are used.
- On OpenVMS I64, IEEE float is the default. IEEE format data is assumed and IEEE floating instructions are used.
- On OpenVMS I64, you can specify the compiler option /FLOAT=D_FLOAT or /FLOAT=G_FLOAT.

When you compile an OpenVMS application that specifies an option to use VAX floating−point on the Intel Itanium architecture, the compiler automatically generates code for converting floating−point formats. Whenever the application performs a sequence of arithmetic operations, this code does the following:

1. Converts VAX floating−point formats to either IEEE single or IEEE double floating−point formats.
2. Performs arithmetic operations in IEEE floating−point arithmetic.
3. Converts the resulting data from IEEE formats back to VAX formats.

VAX floating−point formats have the same number of bits and precision as their equivalent IEEE floating−point formats. For most applications, the conversion process will be transparent. Note that where no arithmetic operations are performed (VAX float fetches followed by stores), conversions will not occur. The code handles such situations as moves.

In a few cases, arithmetic calculations might have different results because of the following differences between VAX and IEEE formats:

- Values of numbers represented
- Rounding rules
- Exception behavior

Matching the default of the native IA64 compilers, the Oracle Rdb and SQL precompiler default floating−point format is now IEEE. The default for the Oracle Rdb and SQL precompilers on OpenVMS

Alpha remains as VAX floating−point format. The Oracle Rdb and SQL precompilers on OpenVMS Alpha also support IEEE floating−point format as an option.

For consistent results and data content, it is important that all portions of the application utilize the same floating−point format. Oracle strongly recommends that the floating−point format is explicitly specified on compiler and pre−compiler commands.

For similar behavior for various floating−point exception conditions, Oracle recommends that customers review and consider compiler IEEE floating−point mode options. In particular, the "FAST" option may provide behavior similar to existing applications on VAX and Alpha systems.

The Oracle Rdb on−disk structures and content and data formats remain unchanged in this release.

Oracle recommends reviewing the white paper "OpenVMS Floating−point Arithmetic on the Intel Itanium Architecture" available from HP.

9.1.2 Features Not Yet Available for OpenVMS I64

The following features or capabilities or components are not currently available to run or are known to not run reliably on OpenVMS I64 with this Oracle Rdb release.

- PL/I compiler and Oracle Rdb PL/I precompilers

9.1.3 Expect Additional Memory Consumption

Due to the increased sizes of image files (especially on Integrity servers) and more aggressive buffering and caching schemes and larger I/O size defaults, you should expect to allocate additional page file quota, working set sizes and buffered I/O byte limit quota when using Oracle Rdb Release 7.2.

In particular, when running on Integrity servers, a page file quota of perhaps three times larger may be required for some applications. It is likely that buffered I/O byte limit quota usage may double when moving to Oracle Rdb Release 7.2 (as maximum I/O sizes for some operations are significantly larger than with prior Oracle Rdb Releases).

9.1.4 Handling of Initialized Overlaid Program Sections on OpenVMS I64

On Alpha and VAX systems, initializations can be done to portions of an overlaid program section. Subsequent initializations to the same portions overwrite initializations from previous modules. The last initialization performed on any byte is used as the final one of that byte for the image being linked. On I64 systems, the ELF (Executable and Linkable Format) object language does not implement the feature of the Alpha and VAX object language which allows the initialization of portions of sections. When an initialization is made, the entire section is initialized. Subsequent initializations of this section may be performed only if the non−zero portions match in value.

Any two overlaid sections are compatible if they are identical in the non−zero values. If they are not compatible, the linker issues the following error:

```
%ILINK-E-INVOVRINI, incompatible multiple initializations for overlaid section
section: <section name>
module: <module name for first overlaid section>
file: <file name for first overlaid section>
module: <module name for second overlaid section>
file: <file name for second overlaid section>
```

In the previous message, the linker lists the first module that contributes a non−zero initialization, and the first module with an incompatible initialization. Note that this is not a full list of all incompatible initializations; it is just the first one the linker encounters.

This particular symptom may be seen with applications using Oracle Rdb when multiple modules attempt to initialize handle values. Only one module may initialize any particular handle. SQL precompilers allow initialization to be controlled with the INITIALIZE_HANDLES keyword of the SQLOPTIONS qualifier.

For more detail on the handling of initialized overlaid sections, see the HP OpenVMS Version 8.2 New Features and Documentation Overview.

9.1.5 Deleted Space in Uniform Areas Not Reclaimed by Other Users

Bug 2551066

In prior releases of Oracle Rdb, when rows were deleted from a table stored in a uniform storage area, other database users would not be aware that space was made available and could extend the storage area when inserting additional rows in the table even though free space was available.

This release of Oracle Rdb introduces a mechanism that allows database users on the same cluster node to share information regarding the availability of free space. When a user chooses a location to store new rows, the location is stored in the database global section so that other users can use that location as a starting point when searching for available space. When a user deletes rows from a table, if the location of the deleted rows is closer to the beginning of the storage area than the last page used for an insert then the starting page for the next insert is updated to the location of the lowest page that had rows deleted.

9.1.6 AIP Entries Cached for Improved Performance

Whenever a table is first accessed within a database attach, Oracle Rdb must look up the description of the table. Some of the table description is stored on disk on pages called Area Inventory Pages (AIPs). These pages are linked together in a special table or "logical area" called RDB$AIP. The AIP pages are sequentially scanned each time it is necessary to find an AIP entry. If a database has many tables defined, then it could take a significant number of I/Os to locate the desired AIP entry in the RDB$AIP list. Prior to this release, the look up was repeated each time a new attach first referenced a table. Thus, applications that often attached to and detached from a database that had many tables defined in it could expend a tremendous amount of I/O constantly reloading AIP entries from disk.

This release introduces an enhancement that, for most applications, should essentially eliminate the RDB$AIP I/O. Now, the first time that a table is referenced the AIP entry is copied into an extended lock value block. (See the OpenVMS Programming Concepts Manual for more information regarding lock value blocks.) Any subsequent reference to the table will find the desired information in the lock value block and thus not need to read the entry from disk. After the most frequently accessed tables have had their AIP entries loaded into lock

value blocks, there should be little, if any, further I/O to the RDB$AIP area.

9.1.7 Improved Rollback Performance

This release of Oracle Rdb introduces additional optimizations for rolling back transactions. These improvements affect the performance of ROLLBACK statements issued by an application and also the database recovery (DBR) process. A summary of the most significant changes are listed below:

- When reading the recovery−unit journal file (RUJ), I/Os are now done using 256 block buffers instead of reading one block at a time as was done in previous versions.
- Multiple buffers are now used to read the journal. While the contents of one buffer are being processed, data is being read into the next buffer asynchronously.
- When writing to the journal, RUJ data is copied directly into the RUJ I/O buffer from the storage area data page instead of being copied into an intermediate buffer and then to the RUJ buffer.
- When reading from the journal, journal entries are processed directly from the RUJ I/O buffer instead of being copied to an intermediate buffer first.
- When rolling back a transaction, the content of the RUJ buffer is scanned to determine what data pages will be rolled back and I/Os are started to those pages immediately. That is, asynchronous prefetches (APF) are issued for pages that will be rolled back. As journal entries are processed, new prefetches are started for subsequent journal entries as soon as buffers are available. This significantly reduces the time spent waiting for I/O completion.
- In previous releases, if a process failed and a DBR was started to recover the user, the DBR would scan the journal to locate the last entry in the journal. For large transactions, the scanning operation could take a considerable amount of time. In this release, the location of the last journal entry is maintained in shared memory. Now, when a DBR process is started it can immediately locate the last entry in the journal without having to scan the journal.

9.1.8 Index Prefetching Performance Improvements

This release of Oracle Rdb introduces an optimization for queries that do index scans to fetch rows from a table. Index scans will now prefetch data pointed to by entries in the index before the application actually requests that the rows be returned. With this optimization, in many instances when an application does request the next row from a result set, the row will already be in an I/O buffer and can be immediately returned to the application.

For example, consider the following table and index definition:

```
CREATE TABLE T1 (C1 INT, C2 INT);
CREATE INDEX I1 ON T1 (C2)
```

The following query will select rows from the table based on a range of values for column C2. Oracle Rdb chooses an index scan retrieval strategy to satisfy the query.

```
SQL> SET FLAGS 'STRATEGY';
SQL> SELECT C1 FROM T1 WHERE C2 > 100 AND C2 < 900000 ORDER BY C2;
  Conjunct        Get     Retrieval by index of relation T1
    Index name  I1 [1:1]
```

When the above query executes, the index node that contains the first C2 value that is greater than 100 is fetched. Then, each entry in the index node that is greater than 100 and less than 900000 is examined and I/O

is started for each data page pointed to by each index entry. Prefetching continues for each entry in the index node until one of the following conditions is met:

- The database ASYNCH PREFETCH DEPTH IS n BUFFERS limit is reached
- The end of the current index node is encountered
- A pointer to a duplicates node is encountered
- The key with the ending scan value (in this example, 900000) is found
- A zig−zag strategy skip is requested

After all possible prefetches have been issued, the first row in the result set is returned to the application. Subsequent fetches for additional rows will find that the I/O request for a needed buffer is already in progress or may even be completed.

Each time that a new entry is requested via the index, if prefetching was stopped due to PREFETCH DEPTH being reached or a new index node being requested, prefetching will resume if that condition is satisfied.

In some applications, the performance improvements from this optimization can be very significant. Large databases that are not readily cached by existing caching products will typically see the greatest improvement in performance.

9.1.9 Performance Improvement for Query with Constant Boolean Predicates

Bug 4205719

The customer reports that the following query where the boolean condition always returns a known value of FALSE uses a full sequential retrieval and becomes very slow on a large table:

```
set flags 'strategy,detail';
select * from resumes where  1 = 2;
Tables:
  0 = RESUMES
Conjunct: 1 = 2
Get     Retrieval sequentially of relation 0:RESUMES
0 row selected
```

Although the condition was always false and 0 rows were returned, Oracle Rdb still performed a sequential table scan. In a database with about 1 million rows, this unnecessary table scan takes a lot of time.

Oracle Rdb has been changed to detect expressions of the following forms and to avoid doing index and table scans if those expressions are non−variable and evaluated as false.

```
WHERE constant−expression
WHERE other−expression AND constant−expression
WHERE constant−expression AND other−expression
```

For example:

```
WHERE 1 = 2
WHERE (1 = 2) AND (LAST_NAME > '')
WHERE (LAST_NAME > '') AND (1 = 2)
```

This does not include expressions that contain host variables, as in the examples below, because host variables are considered to be variable.

```
WHERE :HV = 1
WHERE (:HV1 = 1) AND (LAST_NAME > '')
```

This problem has been corrected in Oracle Rdb Release 7.2.

9.1.10 Index Column Group is Enabled by Default

In prior versions, Index Column Group flag was disabled by default. If this flag is enabled, the Oracle Rdb optimizer will try to find an index that has the same leading columns as the columns of Index Column Group (or Workload Column Group). If a match is found, it uses the index prefix cardinality to calculate the column duplicity and null factors which will help the optimizer to estimate solution costs and cardinalities with higher accuracy.

This flag is now enabled by default.

The following example shows flags that are set by default. This can be overridden using the *RDMS$SET_FLAGS* logical name, or the *SET FLAGS* statement in interactive and dynamic SQL.

```
SQL> show flags

Alias RDB$DBHANDLE:
Flags currently set for Oracle Rdb:
   PREFIX,WARN_DDL,INDEX_COLUMN_GROUP,MAX_SOLUTION
   ,MAX_RECURSION(100),NOBITMAPPED_SCAN
```

RMU Collect Optimizer_Statistics

Oracle strongly recommends that customers use the "RMU /COLLECT OPTIMIZER_STATISTICS" command on databases converted from prior Rdb versions.

9.1.11 No File−System Caching When Writing Database and AIJ Backup Files

It is expected that the disk−based output file from a database or after−image journal backup operation may be relatively large, sequentially accessed and not read in the near future. In order to avoid "polluting" the file system cache and to streamline file write operations, caching by the operating system is now explicitly disabled when writing these files.

9.1.12 Estimation Refinement Rules are Enabled by Default

In prior versions, index estimation was normally performed by descending to the split level in sorted indexes. For more information, please refer to the technical article entitled "Guide to Database Performance and Tuning: Predicate Estimation".

Estimation refinement rules were available to enable greater precision in estimation on indexes of *TYPE IS SORTED RANKED* and to enable estimation on hashed indexes. These rules were enabled using the

REFINE_ESTIMATES flag.

This flag is now enabled by default so that all estimation refinement rules are enabled.

The following example shows flags that are set by default. This can be overridden using the *RDMS$SET_FLAGS* logical name, or the *SET FLAGS* statement in interactive and dynamic SQL.

```
SQL> show flags

Alias RDB$DBHANDLE:
Flags currently set for Oracle Rdb:
   PREFIX,WARN_DDL,INDEX_COLUMN_GROUP,MAX_SOLUTION
   ,MAX_RECURSION(100),REFINE_ESTIMATES(127),NOBITMAPPED_SCAN
```

Notice that the *REFINE_ESTIMATES* flag has a value of 127. Please refer to the technical article above for information on the significance of this value.

The previous behavior can be obtained by setting this flag to zero or negating the flag to disable all refinement rules.

```
SQL> set flags 'refine_estimates(0)'
SQL> show flags

Alias RDB$DBHANDLE:
Flags currently set for Oracle Rdb:
   PREFIX,WARN_DDL,INDEX_COLUMN_GROUP,MAX_SOLUTION
   ,MAX_RECURSION(100),REFINE_ESTIMATES(0),NOBITMAPPED_SCAN
SQL> exit
$ define rdms$set_flags "refine_estimates(0)"
$ sql
SQL> show flags

Alias RDB$DBHANDLE:
Flags currently set for Oracle Rdb:
   PREFIX,WARN_DDL,INDEX_COLUMN_GROUP,MAX_SOLUTION
   ,MAX_RECURSION(100),REFINE_ESTIMATES(0),NOBITMAPPED_SCAN
SQL>exit
$ define rdms$set_flags "norefine_estimates"
$ sql
SQL> show flags

Alias RDB$DBHANDLE:
Flags currently set for Oracle Rdb:
   PREFIX,WARN_DDL,INDEX_COLUMN_GROUP,MAX_SOLUTION
   ,MAX_RECURSION(100),REFINE_ESTIMATES(0),NOBITMAPPED_SCAN
SQL>exit
$ sql
SQL> set flags 'norefine_estimates'
SQL> show flags

Alias RDB$DBHANDLE:
Flags currently set for Oracle Rdb:
   PREFIX,WARN_DDL,INDEX_COLUMN_GROUP,MAX_SOLUTION
   ,MAX_RECURSION(100),REFINE_ESTIMATES(0),NOBITMAPPED_SCAN
```

9.1.13 New LIMIT TO Syntax

This release of Oracle Rdb enhances the LIMIT TO clause by allowing the programmer to skip over some number of delivered rows from the query. For instance, the first row in the result set might be the column headings loaded from a CSV data source loaded by RMU/LOAD/RECORD=FORMAT=DELIMITED and as such should be ignored by queries.

Note

Oracle recommends that the values specified for skip−expression be kept small for performance reasons. These skipped rows are still fetched and processed by the query, they are just not returned to the application.

The following example shows one use of this new feature. This query returns the 100th employee from the EMPLOYEES table.

```
SQL> select last_name, first_name, employee_id
cont> from employees
cont> order by employee_id
cont> limit to 1 skip 99 rows;
 LAST_NAME        FIRST_NAME   EMPLOYEE_ID
 Herbener         James        00471
1 row selected
```

To retrieve the last row in the sorted list, the application programmer could replace the literal value with a subselect that calculates the value. This query also shows the output from the SET FLAGS command for the query strategy. Note the "Skipn" keyword that describes the new SKIP clause.

```
SQL> set flags 'strategy,detail';
SQL> select last_name, first_name, employee_id
cont> from employees
cont> order by employee_id
cont> limit to 1
cont> skip (select count(*)−1 from employees) rows;
Tables:
  0 = EMPLOYEES
  1 = EMPLOYEES
Cross block of 2 entries
  Cross block entry 1
    Aggregate: 0:COUNT (*)
    Index only retrieval of relation 1:EMPLOYEES
      Index name  EMP_EMPLOYEE_ID [0:0]
  Cross block entry 2
    Firstn: 1
    Skipn: <agg0> − 1
    Get     Retrieval by index of relation 0:EMPLOYEES
      Index name  EMP_EMPLOYEE_ID [0:0]
 LAST_NAME        FIRST_NAME   EMPLOYEE_ID
 Herbener         James        00471
1 row selected
SQL>
```

An alternative to this query would be to use ORDER ... DESC and then to use a simple LIMIT 1 ROW clause.

This query finds the statistical median salary.

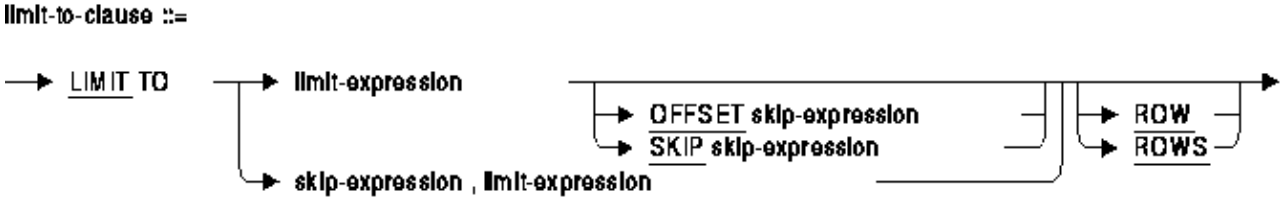
```
SQL> -- select the median salary
SQL> select salary_amount
cont> from salary_history
cont> where salary_end is NULL
cont> order by salary_amount
cont> limit to 1
cont> skip (select count(*)/2
cont>       from salary_history
cont>       where salary_end is NULL);
 SALARY_AMOUNT
    $24,166.00
1 row selected
SQL>
```

The statistical median salary can be compared with the average salary.

```
SQL> -- select the median salary compare with average
SQL> select salary_amount as median_salary,
cont>        (select avg (salary_amount)
cont>         from salary_history
cont>         where salary_end is NULL) as avg_salary edit using SALARY
cont> from salary_history
cont> where salary_end is NULL
cont> order by salary_amount
cont> limit to 1
cont> skip (select count(*)/2
cont>       from salary_history
cont>       where salary_end is NULL);
 MEDIAN_SALARY    AVG_SALARY
    $24,166.00    $31,922.79
1 row selected
SQL>
```

Syntax

The revised SQL syntax for the LIMIT TO clause is:



The syntax variants are supported by different SQL implementations and permit different tools to use this syntax with Oracle Rdb.

Arguments

- limit−expression
  If limit−expression is evaluated to a negative value or zero then no rows are returned from the query and no error is reported.
- skip−expression
  If skip−expression is evaluated to a negative value or zero then no rows are skipped. If the

skip−expression is larger than the rows in the result set then no rows are returned from the query and no error is reported.

If either limit−expression or skip−expression is specified as a numeric literal then it must be an unscaled value. These numeric expressions are converted to BIGINT before executing the query.

Neither limit−expression nor skip−expression may reference columns from the select−expression in which they occur. Only columns of a subselect specified for the limit−expression or skip−expression can be used. See above for examples that use a subselect in the LIMIT TO clause.

9.1.14 Additional %CDD−I−BLRSYNINFO Integrating with CDD

When an Oracle Rdb Release 7.2 database is integrated into a Common Data Dictionary repository, an additional %CDD−I−BLRSYNINFO is generated compared to Oracle Rdb Release 7.1. This additional message is caused by an enhancement to the RDB$DATABASE metadata table so that the value of the RDB$FILE_NAME column can be computed using a function call. The additional informational message is expected and can be ignored.

For example, executing the following SQL statements with Oracle Rdb Release 7.1 would result in a single instance of the %CDD−I−BLRSYNINFO message after the INTEGRATE statement but under Oracle Rdb Release 7.2, it results in two %CDD−I−BLRSYNINFO messages as shown.

```
SQL> CREATE DATABASE FILENAME TEMP;
SQL> DISCONNECT ALL;
SQL> INTEGRATE DATABASE FILENAME TEMP CREATE PATHNAME TEMP;
%CDD−I−BLRSYNINFO, unsupported entity − marked Incomplete
%CDD−I−BLRSYNINFO, unsupported entity − marked Incomplete
```

9.1.15 RMU Unload Record_Definition Accepts TRIM Option

This release of Oracle Rdb adds a TRIM option to the RMU Unload Record_Definition qualifier. The new TRIM option supports three keywords:

- TRAILING − trailing spaces will be trimmed from CHARACTER and CHARACTER VARYING (VARCHAR) data that is unloaded. This is the default setting if only the TRIM option is specified.
- LEADING − leading spaces will be trimmed from CHARACTER and CHARACTER VARYING (VARCHAR) data that is unloaded.
- BOTH − both leading and trailing spaces will be trimmed.

The following example shows the output without using the TRIM option.

```
$ RMU/UNLOAD/RECORD=(FORMAT=DELIMITED) DB$ WORK_STATUS SYS$OUTPUT:
"0","INACTIVE","RECORD EXPIRED"
"1","ACTIVE  ","FULL TIME     "
"2","ACTIVE  ","PART TIME     "
%RMU−I−DATRECUNL,   3 data records unloaded.
```

The results, after adding the TRIM=BOTH option, show that all trailing spaces are removed.

```
$ RMU/UNLOAD/RECORD=(FORMAT=DELIMITED,TRIM=BOTH) DB$ WORK_STATUS SYS$OUTPUT:
"0","INACTIVE","RECORD EXPIRED"
"1","ACTIVE","FULL TIME"
"2","ACTIVE","PART TIME"
%RMU-I-DATRECUNL,   3 data records unloaded.
```

9.1.16 Maximum Page and Buffer Size Increases

Previously, the maximum allowed database buffer size was 64 blocks and the maximum allowed database page size was 32 blocks. These limits have been increased. The current maximum allowed database buffer size is 128 blocks and the maximum allowed database page size is 63 blocks.

Be aware that using larger database buffer sizes will require additional virtual memory and buffered I/O byte count quota.

9.1.17 Various I/O Sizes Increased

Previously, nearly all Oracle Rdb related I/O requests were limited to a maximum of 127 blocks (65,024 bytes). In many areas within Oracle Rdb, this limit has been increased to 256 blocks (131,072 bytes). Writing to and reading from SORT work files is now done with I/O requests up to 1024 blocks (524,288 bytes). These increases should, in some cases, serve to reduce I/O counts, and increase effective I/O through−put at a lower CPU cost (by doing fewer, large I/Os).

This change also will increase the amount of virtual and physical memory required for processes in terms of page file quota and buffered I/O byte count limit.

9.1.18 New Statistics for the Oracle Rdb Executive

Bug 3917094

Several new statistics have been added to a new statistics screen for *RMU/SHOW STATISTICS*.

The following example shows the new screen.

```
Rate: 3.00 Seconds          Rdb Executive Statistics      Elapsed: 00:03:05.26
Page: 1 of 1           DUA0:[PERS.V72]MF_PERSONNEL.RDB;1         Mode: Online
------------------------------------------------------------------------------
statistic.........      rate.per.second.............. total....... average......
name..............      max..... cur..... avg....... count....... per.trans....
queries compiled              18       18        0.3           60          60.0
index scans                   13       13        0.2           42          42.0
 index only                    5        5        0.0           17          17.0
 index full                    5        5        0.0           17          17.0
dynamic optimizer              5        5        0.0           17          17.0
 one abandoned                 0        0        0.0            0           0.0
 all abandoned                 0        0        0.0            0           0.0
```

In this screen, the statistics displayed have the following meanings.

- The "queries compiled" statistic counts the number of times the executive has compiled a request, also called query optimization.

- The "index scans" statistic counts the number of times an index scan is initiated. This statistic accumulates for all index types and for all scan types including direct lookup.
- The "index only" statistic counts the number of scans that are started that are index only scans.
- The "index full" statistic counts the number of index scans started that do not have a lower or upper bound. In this case, the entire index will be scanned. If a key value range is provided by the query that includes all keys in the index, the entire index will be scanned. However, this statistic will not be incremented.
- The "dynamic optimizer" statistic counts the number of times the dynamic optimizer has been invoked.
- The "one abandoned" statistic counts the number of times that the dynamic optimizer abandons a background index scan because it is considered too costly.
- The "all abandoned" statistic counts the number of times that all background indexes have been abandoned and the dynamic optimizer has switched to sequential retrieval.

9.1.19 RMU /SHOW STATISTICS Enhanced Navigation Between Row Caches

Bugs 4727723 and 3738511

Previously, when using the RMU /SHOW STATISTICS "Row Cache Utilization", "Hot Row Information", "Row Cache Status", "Row Cache Queue Length", and "Row Length Distribution" displays, it was difficult to move between displays for multiple row caches.

This problem has been corrected. The "]" and "[" keys can be used to navigate next or previous between row caches on these displays.

9.1.20 RMU SHOW LOCKS /RESOURCE_TYPE Qualifier

Previously, the RMU /SHOW LOCKS command would display all lock resource types. This could sometimes result in a significant amount of output and could make it cumbersome to locate locks for specific types of resources.

This situation has been improved with the /RESOURCE_TYPE=(restyp...) qualifier. When this qualifier is present on the command line, only the specific resource types will be displayed. This permits, for example, only PAGE or RECORD lock types to be selected. This functionality is intended primarily as a debugging tool. Knowledge of the lock types and functionality of Oracle Rdb is assumed. Not all lock types will exist on all systems and versions of Oracle Rdb.

The following keywords are allowed with the /RESOURCE_TYPE qualifier.

Table 9−1 RESOURCE_TYPE keywords

Internal Lock Type Name	Keyword(s)
ACCESS	ACCESS
ACTIVE	ACTIVE
AIJDB	AIJDB

AIJFB	AIJFB
AIJHWM	AIJHWM, AIJ_HIGH_WATER_MARK
AIJLOGMSG	AIJ_LOG_MESSAGE
AIJLOGSHIP	AIJ_LOG_SHIPPING
AIJOPEN	AIJ_OPEN
AIJSWITCH	AIJ_SWITCH
AIJ	AIJ
AIPQHD	AIP
ALS	ALS_ACTIVATION
BCKAIJ	AIJ_BACKUP, BCKAIJ
BCKAIJ_SPD	AIJ_BACKUP_SUSPEND
BUGCHK	BUGCHECK
CHAN	CHAN, FILE_CHANNEL
CLIENT	CLIENT
CLOSE	CLOSE
CLTSEQ	CLTSEQ
CPT	CORRUPT_PAGE_TABLE, CPT
DASHBOARD	DASHBOARD_NOTIFY
DBK_SCOPE	DBKEY_SCOPE
DBR	DBR_SERIALIZATION
DB	DATABASE
FIB	FAST_INCREMENTAL_BACKUP, FIB
FILID	FILID
FRZ	FREEZE
GBL_CKPT	GLOBAL_CHECKPOINT
GBPT_SLOT	GLOBAL_BPT_SLOT
KROOT	KROOT
LAREA	LAREA, LOGICAL_AREA
LOGFIL	LOGFIL
MEMBIT	MEMBIT
MONID	MONID, MONITOR_ID
MONITOR	MONITOR
NOWAIT	NOWAIT
PLN	DBKEY, RECORD, PLN
PNO	PAGE, PNO
QUIET	QUIET
RCACHE	RCACHE
RCSREQUEST	RCS_REQUEST
RCSWAITRQST	RCS_WAIT_REQUEST
REL_AREAS	RELEASE_AREAS
REL_GRIC_REQST	RELEASE_GRIC_REQUEST
RMUCLIENT	RMU_CLIENT

ROOT_AREA	DUMMY_ROOT_AREA
RO_L1	L1_SNAP_TRUNCATION
RTUPB	RTUPB
RUJBLK	RUJBLK
RW_L2	L2_SNAP_TRUNCATION
SAC	SNAP_AREA_CURSOR
SEQBLK	SEQBLK
STAREA	STORAGE_AREA, PAREA
STATRQST	STATISTICS_REQUEST
TRM	TERMINATION
TSNBLK	TSNBLK
UTILITY	UTILITY

The RESOURCE_TYPE qualifier is incompatible with the MODE, LIMIT, LOCK and PROCESS qualifiers.

9.1.21 RMU Command Qualifiers Accept Absolute or Delta Date/Time Specification

The allowed date/time format for several RMU command qualifiers have been extended to include delta as well as absolute times. The following command qualifiers now allow delta or absolute time specifications:

- RMU /SHOW STATISTICS /UNTIL=date/time
- RMU /UNLOAD /AFTER_JOURNAL /BEFORE=date/time
- RMU /UNLOAD /AFTER_JOURNAL /SINCE=date/time
- RMU /CHECKPOINT /UNTIL=date/time
- RMU /BACKUP /TAPE_EXPIRATION=date/time
- RMU /BACKUP /AFTER_JOURNAL /TAPE_EXPIRATION=date/time
- RMU /OPTIMIZE /AFTER_JOURNAL /TAPE_EXPIRATION=date/time
- RMU /BACKUP /AFTER_JOURNAL /UNTIL=date/time
- RMU /RECOVER /UNTIL=date/time
- RMU /DUMP /AFTER_JOURNAL /FIRST=TIME=date/time
- RMU /DUMP /AFTER_JOURNAL /LAST=TIME=date/time

Absolute time includes a specific date or time of day. An absolute date/time has one of the following formats:

- dd−mmm−yyyy
- hh:mm:ss.cc
- dd−mmm−yyyy:hh:mm:ss.cc
- "dd−mmm−yyyy hh:mm:ss.cc"
- BOOT
- LOGIN
- TODAY
- TOMORROW
- YESTERDAY

You can omit any of the trailing fields in the date or time. You can omit any of the fields in the middle of the format as long as you specify the punctuation marks, for example, "−mmm−yyyy hh".

Delta time is an offset from the current time to a time in the future. Delta time has the following format:

- "+[dddd−][hh:mm:ss.cc]"

You can truncate delta time after the hour field. You can also omit any of the fields after the hour field format as long as you specify the punctuation marks.

9.1.22 64−bit Statistics

In prior versions of Oracle Rdb, statistics counters were maintained in 32−bit longword integers. This limited counters to a maximum value of 4,294,967,294. This limit could be exceeded and would cause counters to "wrap" back to zero.

This problem has been corrected. Oracle Rdb statistics counters have been promoted to 64−bit quadword integers. This change effects the binary statistics output file format as well.

Note, however, that most field displays within the RMU /SHOW STATISTICS utility have not been widened and may overflow if the internal counter value exceeds the decimal display width.

9.1.23 Maximum Global Buffer Count Increased

Enhancement Bug 3820284

Prior versions of Oracle Rdb limited the total number of global buffers per database to 524,288. This limit has been relaxed. The maximum global buffer count allowed for Oracle Rdb Release 7.2 is 1,048,576.

9.1.24 Support for WORM (Write Once Read Many) Storage Removed

Prior versions of Oracle Rdb provided support for write−once storage areas on write−once, read−many (WORM) optical disk devices. This support has been removed in Oracle Rdb Release 7.2.

Databases containing storage areas configured as "WRITE ONCE" or "WORM" may not be converted to Oracle Rdb Release 7.2 format nor may they be restored using Oracle Rdb Release 7.2.

The various "WRITE ONCE" or "WORM" keywords and qualifiers in RDO, RMU and SQL are deprecated.

The SQL deprecated message is "%SQL−I−DEPR_FEATURE, Deprecated Feature: WRITE ONCE no longer supported − assuming READ WRITE attribute".

The deprecated message is generated when one specifies a clause:

- WRITE ONCE
- WRITE ONCE (JOURNAL IS ENABLED)
- WRITE ONCE (JOURNAL IS DISABLED)

within one of the following statements:

- ALTER DATABASE ... ADD STORAGE AREA

- ALTER DATABASE ... ALTER STORAGE AREA
- CREATE DATABASE ... CREATE STORAGE AREA
- IMPORT DATABASE ... CREATE STORAGE AREA

For example:

```
SQL> CREATE DATA FILENAME WORM_TEST
cont> CREATE STORAGE AREA WORM_AREA WRITE ONCE;
%SQL-I-DEPR_FEATURE, Deprecated Feature: WRITE ONCE no
longer supported - assuming READ WRITE attribute
```

Use of read−only media continues to be available. A database or storage area may be marked read−only and moved to optical media and accessed in a read−only fashion.

9.1.25 Support for ACE (AIJ Cache on Electronic disk) Removed

Prior versions of Oracle Rdb provided support for a file called an AIJ cache on an electronic disk (also known as ACE) to use as a temporary cache for AIJ write operations. At one point in time, these sorts of devices provided a performance benefit for some classes of applications that heavily used the after−image journal.

With changes in technologies (in particular, improved I/O interfaces and various write−back caching schemes), the benefits of the ACE feature have declined to the point where it is no longer an effective performance advantage. Therefore this support has been removed in Oracle Rdb Release 7.2.

The database attribute "CACHE FILENAME ..." is now ignored by Oracle Rdb. The various related keywords and qualifiers in RMU, RDO and SQL are deprecated.

9.1.26 RMU Support for /DENSITY = SDLT320

Oracle Rdb RMU commands that support the /DENSITY qualifier (ie, RMU/BACKUP, RMU/BACKUP/AFTER_JOURNAL and RMU/OPTIMIZE_AIJ) now support the keyword "SDLT320" for use with SuperDLT320 tape drives.

9.1.27 Sequential Scan Statistics

Bug 3917080

Previously, there was no way to accurately determine the number of strict sequential scans nor the number of DBKEYs returned from those sequential scans.

This problem has been corrected in Oracle Rdb Release 7.2. Two new statistics counters record the number of sequential scans started and the number of DBKEYs returned from those sequential scans. These counters are recorded on a database−wide basis (displayed on the "Record Statistics" screen) and on a per−table basis (displayed on the "Logical Area Statistics" screens).

9.1.28 RDB$SHOVER, RDB$SETVER, SQL$SETVER Temporary Files

Previously, the RDB$SHOVER.COM, RDB$SETVER.COM and SQL$SETVER.COM procedures created temporary files when determining image file identifications. This file creation and deletion activity placed undue burden on the system and also restricted the speed of the procedures.

This problem has been corrected in Oracle Rdb Release 7.2. These procedures no longer create and delete temporary files.

9.1.29 Logical RDM$BIND_RW_TX_CHECKPOINT_ADVANCE Removed

Bug 1584167

Prior to Release 7.1.2, if the logical RDM$BIND_RW_TX_CHECKPOINT_ADVANCE was not defined to be 1, read write transactions that did not make any database modifications would not advance their fast commit checkpoint location. In Release 7.1.2, in response to Bug 2439694, the checkpointing code was restructured such that checkpoints may advance at the end of any transaction, whether or not the transactions made any database modifications. That change made the logical RDM$BIND_RW_TX_CHECKPOINT_ADVANCE no longer necessary. It has been removed.

9.1.30 Backup File Encryption

Oracle Rdb supports encryption of .RBF backup files and .AIJ after−image journal backup files using the new /ENCRYPT qualifier.

Encryption can help increase the level of security on backup data that leaves your security domain or premises. To provide a higher level of security, the backup files are always encrypted with a unique internal key. Even though you may use the same RMU command backing up the same data, the encrypted file differs from the previous backup. This is transparent to the user and the same key is used to decrypt the data.

This feature uses the OpenVMS ENCRYPT component which is included with the operating system starting with OpenVMS V8.2. All encryption algorithms supported by OpenVMS ENCRYPT can be used with RMU. Review the online help and the ENCRYPT documentation for details and supported encryption algorithms. The OpenVMS ENCRYPT component must be installed prior to using the /ENCRYPT qualifier with RMU commands.

Encryption Messages

In order to get the correct message text for encryption messages when running RMU/ENCRYPT, the following file needs to be installed using this command:

```
$INSTALL ADD SYS$MESSAGE:ENCRYPT$_MSG.EXE/OPEN/SHARED
```

The process of encryption takes readable data, called plaintext, and uses a mathematical algorithm to transform the plaintext into an unreadable, unintelligible form, called ciphertext.

To encrypt the plaintext data, the encryption operation requires a key. The key is a variable that controls the encryption operation. The same plaintext, encrypted with different keys, results in different ciphertext. In addition, repeated encryption of the same plaintext with the same key also results in different ciphertext each time.

To gain access to the data in an encrypted file, reverse the encryption process by performing the decryption process. Decryption uses a mathematical encryption algorithm to change ciphertext into the original plaintext.

You can either specify an encryption key directly or predefine a key with DCL−ENCRYPT and use the key name instead in the RMU command line.

Encryption Key

If you cannot remember the encryption key you have effectively lost all data in the encrypted file.

9.1.30.1 Commands Accepting /ENCRYPT

The "/ENCRYPT" qualifier is available for the following commands:

- RMU/BACKUP
- RMU/RESTORE
- RMU/RECOVER
- RMU/DUMP/BACKUP
- RMU/BACKUP/AFTER_JOURNAL
- RMU/DUMP/AFTER_JOURNAL
- RMU/OPTIMIZE/AFTER_JOURNAL

FORMAT=NEW_TAPE

After−image journal backup files have to be in the new tape format (/FORMAT=NEW_TAPE) in order to specify /ENCRYPT.

The /ENCRYPT qualifier has the following format: *Encrypt=([Value=|Name=][,Algorithm=])*

Table 9−2 Encrypt Keywords

Keyword	Description
NAME=key−name	Required if you do not specify key−value. Existing key name previously created and stored in the key storage table with the ENCRYPT /CREATE_KEY command. Specify either the name or the value of a key, but not both.
VALUE=key−value	Required if you do not specify key−name. Interactively defines a value for the key. Specify one of the following: • Character string enclosed in quotation marks (""). • 1 to 243 alphanumeric characters. Dollar signs and underscores are

	valid. Hexadecimal constant using the digits 0 to 9 and A to F. Specify either the name or the value of a key, but not both.
ALGORITHM=DESCBC | DESECB | DESCFB	Algorithm used to encrypt the initialization vector and the key you supply. DESCBC is the default.

Specify a key value as a string or the name of a predefined key that was created with the ENCRYPT /CREATE_KEY command. If no algorithm name is specified, the default is DESCBC. For details on the Value, Name and Algorithm parameters, review the "Encryption for OpenVMS Installation and Reference Manual".

9.1.30.2 Examples

- The following example creates a backup file which is encrypted with the specified key value string and the default encryption algorithm.

```
$ RMU/BACKUP/ENCRYPT=(VALUE="My secret key") -
    MYDB.RDB MYBACKUP.RBF
```

  This backup would be restored using a command similar to this example:

```
$ RMU/RESTORE/ENCRYPT=(VALUE="My secret key") -
    MYBACKUP.RBF
```

- The following example creates a backup file which is encrypted with the specified key name and the default encryption algorithm.

```
$ ENCRYPT /CREATE_KEY /LOG HAMLET -
    "And you yourself shall keep the key of it"
%ENCRYPT-S-KEYDEF, key defined for key name = HAMLET
$ RMU/BACKUP/ENCRYPT=NAME=HAMLET MYDB.RDB MYBACKUP.RBF
```

  This backup would be restored using a command similar to this example:

```
$ RMU/RESTORE/ENCRYPT=NAME=HAMLET MYBACKUP.RBF
```

9.1.31 RMU /POPULATE_CACHE Command /[NO]ONLY_CACHED Qualifier

The RMU /POPULATE_CACHE command allows one or more tables and indexes to be read from the database and stored in caches (if they exist). A new qualifier /[NO]ONLY_CACHED can be used to indicate that all specified tables or indexes are to be read or only those with an associated row cache.

Table 9−3 describes the command qualifiers for the RMU /POPULATE_CACHE command.

Table 9−3 RMU /POPULATE_CACHE Command Qualifiers

Qualifier	Description
/TABLE=table−list	Specifies names of one or more tables to fetch. All rows are fetched from each table. If you list multiple tables, separate the table names with a comma, and enclose the list within parentheses. Wildcard characters "*" and "%" are allowed.
/INDEX=index−list	Specifies names of one or more indexes to fetch. All nodes are fetched from each index. If you list multiple indexes, separate the index names with a comma, and enclose the list within parentheses. Wildcard characters "*" and "%" are allowed.
/LOG	Specifies whether the processing of the command is reported to SYS$OUTPUT. Specify the Log qualifier to request that information about the operation be displayed. If you specify neither /NOLOG nor /LOG, the default is the current setting of the DCL verify switch. (The DCL SET VERIFY command controls the DCL verify switch.)
/[NO]ONLY_CACHED	Specifies if table or index content is to be read only if the table or index has an associated row cache. The default is to read data only from objects that have a cache. If /NOONLY_CACHED is specified, then all data from the specified tables or indexes is read.
/TRANSACTION_TYPE=transaction_mode	Allows you to specify the transaction mode, isolation level, and wait behavior for transactions. Use one of the following keywords to control the transaction mode: • AUTOMATIC − When Transaction_Type=Automatic is specified, the transaction type depends on the current database settings for snapshots (enabled, deferred, or disabled), transaction modes available to this user, and the standby status of the database. Automatic mode is the default. • READ_ONLY − Starts read−only transactions. • WRITE − Starts read−write transactions.

9.1.32 RMU/SHOW LOCKS Includes Time and Node Name

Bug 4761828

The output of the RMU/SHOW LOCKS command has been enhanced to include the current date and time and the system node name in the header line as shown in the following example:

```
$ RMU /SHOW LOCKS
================================================================
  SHOW LOCKS Information at 26−NOV−2005 09:29:01.21 on node RDBI64
================================================================

----------------------------------------------------------------
Resource Name: AIJ journal control
Granted Lock Count: 7,  Parent Lock ID: 180007FA,   Lock Access Mode:
Executive, Resource Type: Global, Lock Value Block: 00000013 00000000
00000000 00000000
```

.
.
.

9.1.33 Default /ROW_COUNT Increased for RMU/UNLOAD and RMU/LOAD

The default value for the /ROW_COUNT qualifier for the RMU/LOAD and RMU/UNLOAD commands has been increased from 50 to 500.

The /ROW_COUNT qualifier specifies that Oracle Rdb buffer multiple rows between the Oracle Rdb server and the RMU Load process. The default is 500 rows; however, this value should be adjusted based on working set size and length of loaded data. Increasing the row count may reduce the CPU cost of the load operation. For remote databases, this may significantly reduce network traffic for large volumes of data because the buffered data can be packaged into larger network packets.

The minimum value you can specify for n is 1. The default row size is the value specified for the Commit_Every qualifier or 500, whichever is smaller.

Chapter 10
Known Problems and Restrictions

This chapter describes problems and restrictions relating to Oracle Rdb and includes workarounds where appropriate.

10.1 Known Problems and Restrictions in All Interfaces

This section describes known problems and restrictions that affect all interfaces.

10.1.1 Patch Required When Using VMS V8.3 and Dedicated CPU Lock Manager

During qualification testing of Oracle Rdb Release 7.2.1 on OpenVMS V8.3 systems, a problem with the use of Extended Lock Value Blocks and the OpenVMS Dedicated CPU Lock Manager feature was discovered.

To avoid this problem, Oracle strongly recommends that customers wishing to use Oracle Rdb and the OpenVMS Dedicated CPU Lock Manager feature with OpenVMS V8.3 install one of the following architecture−specific patch kit (or subsequent replacement if superseded) prior to using Oracle Rdb Release 7.2.1 on OpenVMS V8.3 systems:

- VMS83I_SYS−V0200 (I64)
- VMS83A_SYS−V0100 (Alpha)

10.1.2 ALTER Requires Read−Write Access to Storage Areas

In Oracle Rdb Release 7.2.1, an SQL ALTER TABLE, ALTER STORAGE MAP, TRUNCATE TABLE or ALTER INDEX statement that potentially modifies the on−disk row length requires that all partitions of the object reside in storage areas that allow read−write access. Further, the RDB$SYSTEM storage area is also required to allow read−write access.

The error "RDMS−F−READONLY, data in a read−only storage area may not be accessed for update" will be returned indicating that the storage area being referenced does not allow read−write access as in this example:

```
SQL>ALTER DATA FILE 'PLUGH' ALTER STORAGE AREA U2 READ ONLY;
    .
    .
    .
SQL>ALTER TABLE PARTUNIF ADD COLUMN RDB_TEST1 INT;
%RDB−E−NO_META_UPDATE, metadata update failed
−RDMS−F−READONLY, data in a read−only storage area may
not be accessed for update
```

The restriction that the modified object reside in storage areas that allow read−write access is expected to be lifted in a future Oracle Rdb release.

10.1.3 AIP Length Not Set by ALTER TABLE for Unmapped Tables

This release of Oracle Rdb adds support for updating the length in the AIP (see note Section 6.1.21).

However, ALTER TABLE for tables that do not have a STORAGE MAP with a STORE clause are not currently having the length updated in the AIP.

The following example shows the problem (no LOGMODVAL message in the log output).

```
SQL> set flags 'stomap_stats';
SQL>
SQL> create table T (a integer);
SQL>
SQL> alter table T
cont>     add column b varchar(230);
~As: reads: async 0 synch 8, writes: async 10 synch 0
SQL>
SQL> commit;
SQL>
SQL> create storage map T_MAP
cont>     for T
cont>     store in RDB$SYSTEM;
~As: create storage map "T_MAP"
~As: Table "T" (sys=0, rest=0, tmptbl=0)
~As: creating storage mapping routine T_MAP (columns=0)
~As: creating system module RDB$STORAGE_MAPS
SQL>
SQL> alter table T
cont>     add column c timestamp(2);
~As: reads: async 0 synch 15, writes: async 11 synch 0
SQL>
SQL> commit;
%RDMS-I-LOGMODVAL,     modified record length to 252
%RDMS-W-REBUILDSPAMS, SPAM pages should be rebuilt for logical area T
~As unlocking table "T" (PU -> PR)
```

To workaround this problem, you can add a storage map to the table as shown in the example above, or use the new RMU Set AIP command as shown in the following example.

```
SQL> set flags 'stomap_stats';
SQL>
SQL> create table T (a integer);
SQL>
SQL> alter table T
cont>     add column b varchar(230);
~As: reads: async 0 synch 8, writes: async 10 synch 0
SQL>
SQL> commit;
$ define/user rdms$set_flags "stomap_stats"
$ rmu/set aip abc t/length
%RDMS-I-LOGMODVAL,     modified record length to 244
%RDMS-W-REBUILDSPAMS, SPAM pages should be rebuilt for logical area T
```

This problem will be corrected in the next Oracle Rdb Release after 7.2.1.

10.1.4 SQL/Services Executor Loops Consuming 99% CPU

Bugs 4401924 and 5353228

After upgrading to SQL/Services 7.1.5.9.1 with Rdb 7.1.4.3.1 and later versions, some applications occasionally experienced a SQL/Services executor which entered a tight loop consuming a very high

percentage of the CPU until it was stopped. This problem typically happens immediately after an executor begins servicing a new client and severely degrades the performance of everything else on the VMS node where it occurs. The only way to clear the problem is to STOP/ID the looping executor process.

The proximate cause of the looping behavior is a corrupted cached metadata queue in Rdb's Dynamic SQL image. While it is not known what causes this corruption, code has been added in Rdb Release 7.2.1 to detect the condition rather than looping. This code will report diagnostic data in the executor log and then bugcheck, causing the process to terminate. If an executor terminates because a corrupt metadata queue has been detected, the log will include a potentially long list of metadata queue entries similar to the following:

```
-----Visible parsyms-----
PARSYM block at address: 00E8C040
Serial number: 0000017C = 380
PARSYM[PARSYM$A_NAME]  - address 00E8C030, contents "EMPLOYEES"
PARSYM$A_STORED_NAME - address 00E8D1C0, contents "EMPLOYEES"
PARSYM$A_SYS_QUERY_NAME - address 00E8D1C0, contents "EMPLOYEES"
PARSYM$Q_STMT_QUE       (offset is 000000C0)    The QUE is not used.
PARSYM$Q_MSG_QUE        (offset is 000000C8)    The QUE is not used.
PARSYM$Q_QUE    (offset is 00000004)
    QUE values are fwd = 00E874A4, back = 00E8BF34
    Next entry is 00E874A0/00E874A4, back = 00E8BF30/00E8BF34
PARSYM$Q_SYMBOL_TABLE_QUE       (offset is 0000000C)
    QUE values are fwd = 00E8C23C, back = 00E100B8
    Next entry is 00E8C230/00E8C23C, back = 00E100AC/00E100B8
PARSYM$Q_DSC -
    class 0 dtype = Z {0, %x00}     length 0        pointer 00000000
PARSYM$L_CONTEXT 00E8C150       PARSYM$A_PARRSE 00000000
PARSYM$A_VAR_DEFAULT 00000000   PARSYM$A_PARASS 00000000
PARSYM$A_MSG 00E8C120   PARSYM$A_SUBQ_PARSYM 00000000
PARSYM$A_OUTPUT_PARSYM: 00000000        PARSYM$V_BLOB <False>
PARSYM$V_BLOB_ID_RETRIEVED <False>      PARSYM$V_BLOB_ID_QUEUED <False>
PARSYM$V_CVT_DX_DX <False>      PARSYM$V_COLUMN_NOT_NULL <False>
PARSYM$V_DONE <False>   PARSYM$V_VIEW <False>   PARSYM$V_EXIST_ASSUMED <False>
PARSYM$V_DROP_PENDING <False>   PARSYM$W_ID: 34 PARSYM$A_EDIT_STRING: 00000000
PARSYM$A_QRY_HDR: 00000000      PARSYM$A_REF_MSG_PARSYM: 00000000
PARSYM$B_TYPE is        TABLE
PARSYM$V_TABLE_FORWARD <False>
```

Following the cached metadata reporting, the diagnostic reporting will output a list of entries about database connections for the process.

Normally these entries will appear in the SQL/Services log for the executor reporting the corrupted metadata queue. Immediately after the metadata and connection information, a message similar to one of the following will appear:

```
%SQL-F-BUGCHK, There has been a fatal error. Please contact your Oracle support
representative. SQL$CTX - Circular metadata queue

%SQL-F-BUGCHK, There has been a fatal error. Please contact your Oracle support
representative. SQL$CTX - Self-loop in metadata queue entry
```

The SQL/Services log for the failing executor should be forwarded to Oracle Rdb Support. The log should also contain a %DBS-F-BUGCHECK message noting the location of a SQL/Services bugcheck dump. This should be also be forwarded to Oracle Rdb Support.

There is no workaround for this problem.

10.1.5 SQL Module or Program Fails with %SQL−F−IGNCASE_BAD

Bug 2351258

A SQL Module or Pre−compiled SQL program built with Rdb 6.1 or earlier may fail when running under Rdb 7.2 if the program submits queries that involve certain kinds of character operations on parameters in the queries. For example, a LIKE operator in the WHERE clause of a SQL statement requires SQL to look for character− or string−matching wildcard characters. Another example is the use of IGNORE CASE which causes SQL to equivalence upper and lower case characters for the character set in use.

The following example shows a portion of a SQL module language program that queries a PERSONNEL database.

```
DECLARE MANL_NAME_LIST CURSOR FOR
 SELECT DISTINCT E.LAST_NAME,E.FIRST_NAME,J.JOB_CODE,J.DEPARTMENT_CODE,E.CITY
FROM   DB1_HANDLE.EMPLOYEES E,DB1_HANDLE.JOB_HISTORY J
 WHERE J.EMPLOYEE_ID = E.EMPLOYEE_ID
   AND E.STATUS_CODE = STATUS_CODE
   AND E.CITY LIKE CITYKEY IGNORE CASE
   ORDER BY E.EMPLOYEE_ID DESC, E.LAST_NAME DESC

PROCEDURE SQL_OPN_NAME_LIST
SQLCODE
CITYKEY       CHAR(20)
STATUS_CODE   CHAR(1);
OPEN MANL_NAME_LIST;
```

If the SQL Module containing the code above is compiled and linked into an executable using a pre−7.0 version of Rdb, it will run properly against that version. However if the same program is run in an Rdb 7.2 environment, a call to the SQL_OPN_NAME_LIST procedure will return a SQLCODE of −1. The RDB$MESSAGE_VECTOR will contain a code associated with the following message:

```
%SQL−F−IGNCASE_BAD, IGNORE CASE not supported for character set
```

To workaround this problem, re−link the program using a 7.2 version of SQL$INT.EXE and/or SQL$USER.OLB.

10.1.6 External Routine Images Linked with PTHREAD$RTL

The OpenVMS Guide to the POSIX Threads Library describes that it is not supported to dynamically activate the core run−time library shareable image PTHREAD$RTL. Oracle has found in testing that a shareable image supplied for use as an External Routine that is linked with PTHREAD$RTL can be expected to cause a hang during dynamic image activation on OpenVMS I64 systems. This problem has not been observed on OpenVMS Alpha systems.

To avoid this problem in any case where the shareable image used for an Rdb External Routine is linked with PTHREAD$RTL, the main program image must likewise be linked with PTHREAD$RTL. This requirement applies to customer built application main programs as well as the main interactive SQL image.

The shareable image RDB$NATCONN_FUNC72.EXE supplied with OCI Services for Oracle Rdb (part of SQL/Services) is one such shareable image that is linked with PTHREAD$RTL. Customer built applications

that utilize External Routines from the RDB$NATCONN_FUNC72.EXE image must ensure that the main image is linked with PTHREAD$RTL. The external routines that a user may call that use functions from RDB$NATCONN_FUNC72.EXE include:

- TO_CHAR
- TO_NUMBER
- TO_DATE

You can use the OpenVMS command ANALYZE/IMAGE to determine whether an image depends upon PTHREAD$RTL. For more information, see the OpenVMS documentation.

10.1.7 SQL Procedure External Location Should Be Upper Case

Bug 4722422

When using External Routines, it is important that all declarations for the same shareable image use the exact same strings for the image file specification. Failure to use the same string content may result in multiple copies of the image being activated or failure to correctly call the external routine.

The "ALTER FUNCTION ... LOCATION" command can be used to alter the existing function location string without having to drop and recreate the function.

The following example shows the same string for the EXTERNAL LOCATION specifications:

```
create procedure sys$asctim(
    out :timlen smallint by reference,
    out :timbuf char(23) by descriptor,
    in  :timadr date vms by reference,
    in  :cvtflag integer by value);
    external location 'SYS$SHARE:SYS$PUBLIC_VECTORS.EXE'
    language general general parameter style;

create procedure sys$gettim(
    in  :timadr date vms by reference);
    external location 'SYS$SHARE:SYS$PUBLIC_VECTORS.EXE'
    language general general parameter style;
```

10.1.8 Using Databases from Releases Earlier than V7.0

You cannot convert or restore databases earlier than the Oracle Rdb V7.0 format directly to Oracle Rdb V7.2 format. The RMU Convert command for Oracle Rdb V7.2 supports conversions from Oracle Rdb V7.0 and V7.1 format databases only. If you have an Oracle Rdb V3.0 through V6.1 format database, you must convert it to at least Oracle Rdb V7.0 format and then convert it to Oracle Rdb V7.2 format. For example, if you have a V4.2 format database, you must convert it first to at least Oracle Rdb V7.0 format, then convert it to Oracle Rdb V7.2 format.

If you attempt to convert or restore a database that is prior to Oracle Rdb V7.0 format directly to Oracle Rdb V7.2 format, Oracle RMU generates an error.

10.1.9 Partitioned Index with Descending Column and Collating Sequence

Bug 2797443

A known problem exists in which a query can return wrong results (number of rows returned is incorrect). This can happen on a table that has a multi−column, partitioned index in which one of the columns is sorted in descending order and the column has an associated collating sequence.

The following example can be used to demonstrate the problem.

```
$ sql$
create database file mf_collating.rdb alloc 10
  collating sequence french french
  create storage area area1 alloc 10
  create storage area area2 alloc 10
  create storage area area3 alloc 10;
create table tab1 (id tinyint, r3 char (3));
insert into tab1 (id, r3) values (1, 'a');
insert into tab1 (id, r3) values (1, 'b');
insert into tab1 (id, r3) values (1, 'f');
create index y3 on tab1 (id asc, r3 desc)
       store using (id, r3)
       in area1 with limit of (1, 'k')
       in area2 with limit of (1, 'e')
       otherwise in area3 ;
commit;

set flags 'strategy';

! Here is a query that returns the correct rows using sequential rather
! than indexed access.

select id, r3 from tab1 where id = 1 and r3 <= 'e'
  optimize for sequential access;
Conjunct        Get     Retrieval sequentially of relation TAB1
   ID   R3
    1   a
    1   b
2 rows selected

! Here is the same query without the sequential access restriction.
! Note in the query strategy that index Y3 is used for data retrieval.
! This query ought to (but does not) return the same set of rows as
! for the sequential access query.

select id, r3 from tab1 where id = 1 and r3 <= 'e';
Leaf#01 FFirst TAB1 Card=3
  BgrNdx1 Y3 [2:1] Fan=16
0 rows selected
```

10.1.10 Domain−Qualified TCP/IP Node Names in Distributed Transactions

Bug 3735144

When using TCP/IP for Oracle Rdb remote connections, distributed transactions involving databases on nodes which are not on the same subnet may not work.

Remote Rdb has the capability to make remote connections via TCP/IP in lieu of DECnet. (See the Oracle Rdb OpenVMS Installation and Configuration Guide for how to set this up.) However, distributed transactions involving remote databases connected to via TCP/IP have been difficult. This is because Rdb relies on OpenVMS DECdtm for distributed transaction support and DECdtm requires DECnet for off−node communication. (This is an OpenVMS and not an Rdb restriction. Contact Hewlett−Packard OpenVMS Support for more details.)

OpenVMS provides a capability to run DECnet over TCP/IP so that OpenVMS services which require DECnet (like DECdtm) can operate in an environment where a TCP/IP network is used as the communications backbone. This capability allows DECdtm (and hence Rdb) to manage distributed transactions via TCP/IP. (See HP's OpenVMS DECnet−Plus documentation set for how to configure and use this capability.)

However, for a transaction involving a remote database, Rdb only provides the SCSNODE name of the remote node to DECdtm. For example, consider the following SQL attaches to two remote databases using TCP/IP:

```
SQL> attach 'alias db1 filename node1.a.b.c::db_root:db1 user ''me'' using
''pw''';
SQL> attach 'alias db2 filename node1.a.b.c::db_root:db2 user ''me'' using
''pw''';
```

In the above example, Rdb can successfully connect to both remote databases using the TCP/IP address "node1.a.b.c." but when multiple databases are attached, Rdb implicitly uses distributed transactions via DECdtm. Since Rdb only passes DECdtm the SCSNODE name retrieved from the RDBSERVERnn at the other end of the connection, DECdtm does not, in general, have the information it needs to resolve the remote reference. It will only be able to do so if the SCSNODE name and the TCP/IP node name are the same and the local node is on the same subnet (i.e. ".a.b.c" in the example). Otherwise, after the second attach is made, the following error message will be received as soon as a transaction is started:

```
SQL> set trans read write;
%RDB−F−SYS_REQUEST_CAL, error from system services request − called from 100001
−RDB−E−DECDTMERR, DECdtm system service call error
−IPC−E−BCKTRNSFAIL, failure on the back translate address request
```

There are three potential workarounds:

- If distributed transactions are unimportant to the application, they can be disabled by defining the logical name SQL$DISABLE_CONTEXT to TRUE. Rdb will then not call DECdtm and the node name resolution problem will not be seen. However, it will be the problem of the application to maintain database integrity in the event that a commit succeeds on one database and not on another. See the Rdb Guide to Distributed Transactions for more information.
- If all the nodes involved in the distributed transaction are in the same domain, then TCP/IP can resolve the node with only the first part of the node provided that the SCSNODE name is identical to it. In the example above, this would mean that the remote node had an SCSNODE name of "NODE1" and that the local node was on TCP/IP subnet ".a.b.c".
- It may also be possible to define a DNS/BIND alias name for the remote node's SCSNODE name to the local node's TCP/IP database. This should allow the SCSNODE name passed by Rdb Dispatch to be translated successfully. For example, assuming HP TCP/IP Services for OpenVMS is the TCP/IP

protocol stack then a command like the following could be used on the local node:

```
$ TCP SET HOST NODE1.A.B.C/address=nnn.nnn.nnn.nnn/alias=NODE1_SCS
```

Where "nnn.nnn.nnn.nnn" is the IP address and "NODE1_SC" the OpenVMS SCSNODE name of the remote node. See the HP DECnet−Plus documentation set for more information on how to maintain TCP/IP domain databases.

10.1.11 Some SQL Dialect−required Warnings Not Delivered

Bugs 3651847 and 4532451

The required warnings (information codes) for such things as rows eliminated for nulls (%RDB−I−ELIM_NULL) and string truncation (%RDB−I−TRUN_RTRV) are not being returned for singleton SELECT and singleton UPDATE statements (for example, statements that return a single row using the INTO clause). To demonstrate with a PERSONNEL database, use the following interactive SQL commands:

```
SQL> set dialect 'sql92';
SQL> attach 'filename sql$database';
SQL>
SQL> ! Force a row to contain NULL for SALARY_AMOUNT
SQL> update salary_history
cont> set salary_amount = NULL
cont> where employee_id = '00471'
cont>   and salary_end = date vms'20−Aug−1981';
1 row updated
SQL>
SQL> declare :avg_sal integer(2);
SQL>
SQL> ! No informational generated (but is expected)
SQL> select avg(salary_amount) into :avg_sal
cont> from salary_history where employee_id = '00471'
cont>  and salary_end >= date vms'1−AUG−1970';
SQL> show sqlca
SQLCA:
        SQLCAID:        SQLCA           SQLCABC:        128
        SQLCODE:        0
        SQLERRD:        [0]: 0
                        [1]: 0
                        [2]: 1
                        [3]: 0
                        [4]: 0
                        [5]: 0
        SQLWARN0:       0       SQLWARN1:       0       SQLWARN2:       0
        SQLWARN3:       0       SQLWARN4:       0       SQLWARN5:       0
        SQLWARN6:       0       SQLWARN7:       0
        SQLSTATE:       00000
SQL> print :avg_sal;
        AVG_SAL
       60893.86
SQL>
SQL> ! Non singleton query returns correct informational
SQL> select avg(salary_amount)
cont> from salary_history where employee_id = '00471'
```

```
cont>  and salary_end >= date vms'1-AUG-1970';

  6.089385714285714E+004
1 row selected
%RDB-I-ELIM_NULL, null value eliminated in set function
SQL> show sqlca
SQLCA:
        SQLCAID:        SQLCA           SQLCABC:        128
        SQLCODE:        1003
        SQLERRD:        [0]: 0
                        [1]: 0
                        [2]: 1
                        [3]: 0
                        [4]: 0
                        [5]: 0
        SQLWARN0:       0       SQLWARN1:       0       SQLWARN2:       0
        SQLWARN3:       0       SQLWARN4:       0       SQLWARN5:       0
        SQLWARN6:       0       SQLWARN7:       0
        SQLSTATE:       01003
%RDB-I-ELIM_NULL, null value eliminated in set function
SQL>
SQL> rollback;
```

Since there is a row in the SALARY_HISTORY table with a NULL in SALARY_AMOUNT, the set function AVG should report an informational message (and return a special warning level SQLSTATE/SQLCODE value).

```
%RDB-I-ELIM_NULL, null value eliminated in set function
```

10.1.12 ILINK−E−INVOVRINI Error on I64

When linking an application with multiple modules, the following error message may be returned:

```
%ILINK-E-INVOVRINI, incompatible multiple initializations for overlaid section
        section: VMSRDB
        module: M1
        file: DKA0:[BLD]M1.OBJ;1
        module: M2
        file: DKA0:[BLD]SYS.OLB;1
```

On I64 systems, it is not allowed to have a program section that attempts to be initialized a subsequent time where the non−zero portions of the initializations do not match. This is a difference from OpenVMS Alpha and VAX systems where the linker permitted such initializations.

If the modules specified are SQL module language or precompiler produced, the application build procedures usually need to be modified. Typically, the solution is to initialize the database handles in only one of the modules. The SQLMOD command line qualifiers /NOINITIALIZE_HANDLES and /INITIALIZE_HANDLES are used to specify whether or not alias definitions are coerced into alias references.

10.1.13 New Attributes Saved by RMU/LOAD Incompatible With Prior Versions

Bug 2676851

To improve the behavior of unloading views, Oracle Rdb Release 7.1.2 changed the way view columns were unloaded so that attributes for view computed columns, COMPUTED BY and AUTOMATIC columns were saved. These new attributes are not accepted by prior releases of Oracle Rdb.

The following example shows the reported error trying to load a file from V7.1.2 under V7.1.0.4.

```
%RMU-F-NOTUNLFIL, Input file was not created by RMU UNLOAD
%RMU-I-DATRECSTO,   0 data records stored.
%RMU-F-FTL_LOAD, Fatal error for LOAD operation at 21-OCT-2003 16:34:54.20
```

You can workaround this problem by using the /RECORD_DEFINITION qualifier and specifying the FORMAT=DELIMITED option. However, this technique does not support LIST OF BYTE VARYING column unloading.

10.1.14 SYSTEM−F−INSFMEM Fatal Error With SHARED MEMORY IS SYSTEM or LARGE MEMORY IS ENABLED in Galaxy Environment

When using the GALAXY SUPPORT IS ENABLED feature in an OpenVMS Galaxy environment, a *%SYSTEM−F−INSFMEM, insufficient dynamic memory error* may be returned when mapping record caches or opening the database. One source of this problem specific to a Galaxy configuration is running out of Galaxy Shared Memory regions. For Galaxy systems, GLX_SHM_REG is the number of shared memory region structures configured into the Galaxy Management Database (GMDB).

While the default value (for OpenVMS versions through at least V7.3−1) of 64 regions might be adequate for some installations, sites using a larger number of databases or row caches when the SHARED MEMORY IS SYSTEM or LARGE MEMORY IS ENABLED features are enabled may find the default insufficient.

If a *%SYSTEM−F−INSFMEM, insufficient dynamic memory* error is returned when mapping record caches or opening databases, Oracle Corporation recommends that you increase the GLX_SHM_REG parameter by 2 times the sum of the number of row caches and number of databases that might be accessed in the Galaxy at one time. As the Galaxy shared memory region structures are not very large, setting this parameter to a higher than required value does not consume a significant amount of physical memory. It also may avoid a later reboot of the Galaxy environment. This parameter must be set on all nodes in the Galaxy.

Galaxy Reboot Required

Changing the GLX_SHM_REG system parameter requires that the OpenVMS Galaxy environment be booted from scratch. That is, all nodes in the Galaxy must be shut down and then the Galaxy reformed by starting each instance.

10.1.15 Oracle Rdb and OpenVMS ODS−5 Volumes

OpenVMS Version 7.2 introduced an Extended File Specifications feature, 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" directory trees.

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.

Oracle does support Oracle Rdb database file components on ODS−5 volumes provided that all of these files and directories used by Oracle Rdb strictly follow the ODS−2 file and directory name conventions. In particular, all file names must be specified entirely in uppercase and "special" characters in file or directory names are forbidden.

10.1.16 Optimization of Check Constraints

Bug 1448422

When phrasing constraints using the "CHECK" syntax, a poorer strategy can be chosen by the optimizer than when the same or similar constraint is phrased using referential integrity (PRIMARY and FOREIGN KEY) constraints.

For example, I have two tables T1 and T2, both with one column, and I wish to ensure that all values in table T1 exist in T2. Both tables have an index on the referenced field. I could use a PRIMARY KEY constraint on T2 and a FOREIGN KEY constraint on T1.

```
SQL> alter table t2 alter column f2 primary key not deferrable;
SQL> alter table t1 alter column f1 references t2 not deferrable;
```

When deleting from the PRIMARY KEY table, Rdb will only check for rows in the FOREIGN KEY table where the FOREIGN KEY has the deleted value. This can be seen as an index lookup on T1 in the retrieval strategy.

```
SQL> delete from t2 where f2=1;
Get     Temporary relation      Retrieval by index of relation T2
  Index name  I2 [1:1]
Index only retrieval of relation T1
  Index name  I1 [1:1]
%RDB-E-INTEG_FAIL, violation of constraint T1_FOREIGN1 caused operation to fail
```

The failure of the constraint is not important. What is important is that Rdb efficiently detects that only those rows in T1 with the same values as the deleted row in T2 can be affected.

It is necessary sometimes to define this type of relationship using CHECK constraints. This could be necessary because the presence of NULL values in the table T2 precludes the definition of a primary key on that table. This could be done with a CHECK constraint of the form:

```
SQL> alter table t1 alter column f1
cont>   check (f1 in (select * from t2)) not deferrable;
SQL> delete from t2 where f2=1;
Get     Temporary relation      Retrieval by index of relation T2
  Index name  I2 [1:1]
Cross block of 2 entries
  Cross block entry 1
    Index only retrieval of relation T1
      Index name  I1 [0:0]
  Cross block entry 2
    Conjunct        Aggregate-F1    Conjunct
    Index only retrieval of relation T2
      Index name  I2 [0:0]
%RDB-E-INTEG_FAIL, violation of constraint T1_CHECK1 caused operation to fail
```

The cross block is for the constraint evaluation. This retrieval strategy indicates that to evaluate the constraint, the entire index on table T1 is being scanned and for each key, the entire index in table T2 is being scanned. The behavior can be improved somewhat by using an equality join condition in the select clause of the constraint:

```
SQL> alter table t1 alter column f1
cont>   check (f1 in (select * from t2 where f2=f1)) not deferrable;

or:

SQL> alter table t1 alter column f1
cont>   check (f1=(select * from t2 where f2=f1)) not deferrable;
```

In both cases the retrieval strategy will look like this:

```
SQL> delete from t2 where f2=1;
Get     Temporary relation      Retrieval by index of relation T2
  Index name  I2 [1:1]
Cross block of 2 entries
  Cross block entry 1
    Index only retrieval of relation T1
      Index name  I1 [0:0]
  Cross block entry 2
    Conjunct        Aggregate-F1    Conjunct
    Index only retrieval of relation T2
      Index name  I2 [1:1]
%RDB-E-INTEG_FAIL, violation of constraint T1_CHECK1 caused operation to fail
```

While the entire T1 index is scanned, at least the value from T1 is used to perform an index lookup on T2.

These restrictions result from semantic differences in the behavior of the "IN" and "EXISTS" operators with respect to null handling, and the complexity of dealing with non−equality join conditions.

To improve the performance of this type of integrity check on larger tables, it is possible to use a series of triggers to perform the constraint check. The following triggers perform a similar check to the constraints above.

```
SQL> create trigger t1_insert after insert on t1
cont>  when (not exists (select * from t2 where f2=f1))
cont>    (error) for each row;
SQL> create trigger t1_update after update on t1
cont>  when (not exists (select * from t2 where f2=f1))
cont>    (error) for each row;
```

```
SQL> ! A delete trigger is not needed on T1.
SQL> create trigger t2_delete before delete on t2
cont>  when (exists (select * from t1 where f1=f2))
cont>    (error) for each row;
SQL> create trigger t2_modify after update on t2
cont>  referencing old as t2o new as t2n
cont>  when (exists (select * from t1 where f1=t2o.f2))
cont>    (error) for each row;
SQL> ! An insert trigger is not needed on T2.
```

The strategy for a delete on T2 is now:

```
SQL> delete from t2 where f2=1;
Aggregate-F1    Index only retrieval of relation T1
  Index name  I1 [1:1]
Temporary relation      Get     Retrieval by index of relation T2
  Index name  I2 [1:1]
%RDB-E-TRIG_INV_UPD, invalid update; encountered error condition defined for
trigger
-RDMS-E-TRIG_ERROR, trigger T2_DELETE forced an error
```

The trigger strategy is the index only retrieval displayed first. You will note that the index on T1 is used to examine only those rows that may be affected by the delete.

Care must be taken when using this workaround as there are semantic differences in the operation of the triggers, the use of "IN" and "EXISTS", and the use of referential integrity constraints.

This workaround is useful where the form of the constraint is more complex, and cannot be phrased using referential integrity constraints. For example, if the application is such that the value in table T1 may be spaces or NULL to indicate the absence of a value, the above triggers could easily be modified to allow for these semantics.

10.1.17 Carryover Locks and NOWAIT Transaction Clarification

In NOWAIT transactions, the BLAST (Blocking AST) 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 carryover locks. There can be a delay before the transactions with carryover locks detect the presence of the NOWAIT transaction and release their carryover locks. You can detect this condition by examining the stall messages. If the "Waiting for NOWAIT signal (CW)" stall message appears frequently, the application is probably experiencing a decrease in performance, and you should consider disabling the carryover lock behavior.

10.1.18 Unexpected Results Occur During Read−Only Transactions on a Hot Standby Database

When using Hot Standby, it is typical to use the standby database for reporting, simple queries, and other read−only transactions. If you are performing these types of read−only transactions on a standby database, be

sure you can tolerate a READ COMMIT level of isolation. This is because the Hot Standby database might be updated by another transaction before the read−only transaction finishes, and the data retrieved might not be what you expected.

Because Hot Standby does not write to the snapshot files, the isolation level achieved on the standby database for any read−only transaction is a READ COMMITED transaction. This means that nonrepeatable reads and phantom reads are allowed during the read−only transaction:

- Nonrepeatable read operations: Allows the return of different results within a single transaction when an SQL operation reads the same row in a table twice. Nonrepeatable reads can occur when another transaction modifies and commits a change to the row between transactions. Because the standby database will update the data when it confirms a transaction has been committed, it is very possible to see an SQL operation on a standby database return different results.
- Phantom read operations: Allows the return of different results within a single transaction when an SQL operation retrieves a range of data values (or similar data existence check) twice. Phantoms can occur if another transaction inserted a new record and committed the insertion between executions of the range retrieval. Again, because the standby database may do this, phantom reads are possible.

Thus, you cannot rely on any data read from the standby database to remain unchanged. Be sure your read−only transactions can tolerate a READ COMMIT level of isolation before you implement procedures that read and use data from a standby database.

10.1.19 Both Application and Oracle Rdb Using SYS$HIBER

In application processes that use Oracle Rdb and the $HIBER system service (possibly through RTL routines such as LIB$WAIT), the application must ensure 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) feature is enabled.

The use of the $WAKE system service by Oracle Rdb 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 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.

```
OWN WAKEFLG : VOLATILE; ! Volatile to force memory fetch

ROUTINE TIMER_WAIT:
    BEGIN
    WAKEFLG = FALSE ! Clear timer flag

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

```
    END

ROUTINE TIMER_AST:
    BEGIN
    WAKEFLG = TRUE ! Set flag indicating timer expired
    STAT = SYS$WAKE () ! Wake the main-line code
    IF STAT <> SS$_NORMAL THEN LIB$SIGNAL (STAT)
    END
```

The LIB$K_NOWAKE flag can be specified when using the OpenVMS LIB$WAIT routine 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.

10.1.20 Row Cache Not Allowed While Hot Standby Replication is Active

The row cache feature may not be enabled on a hot standby database while replication is active. The hot standby feature will not start if row cache is enabled.

This restriction exists because rows in the row cache are accessed via logical dbkeys. However, information transferred to the standby database via the after image journal facility only contains physical dbkeys. Because there is no way to maintain rows in the cache via the hot standby processing, the row cache must be disabled when the standby database is open and replication is active.

A new command qualifier, ROW_CACHE=DISABLED, has been added to the RMU Open command. To open the hot standby database prior to starting replication, use the ROW_CACHE=DISABLED qualifier on the RMU Open command.

10.1.21 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. 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 a 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 shareable image. SQL IMPORT and RMU Load operations do, however, call the OpenVMS SORT run−time library.

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

SORT does not directly consider the processes 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 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 the new pages are faulted in. Once the sort operation completes and SORT returns back to Oracle Rdb, 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. Since that 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 for Oracle Rdb sort operations. When the operation can not be done in the available memory, SORT uses temporary disk files to hold the data as it is being sorted. The Oracle Rdb7 Guide to Database Performance and Tuning contains more detailed information about sort work files.

The logical name RDMS$BIND_SORT_WORKFILES specifies how many work files sort is to use if work files are required. The default is 2 and the maximum number is 36. The work files can be individually controlled by the SORTWORKn logical names (where n ranges from "0" through "Z"). 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 36 logical names, "SORTWORK0" through "SORTWORKZ".

Normally, SORT places work files in the your SYS$SCRATCH directory. By default, SYS$SCRATCH is the same device and directory as the SYS$LOGIN location. Spreading the I/O load over multiple disks and/or controllers improves efficiency as well as performance by taking advantage of more system resources and helps prevent disk I/O bottlenecks. Specifying that a your work files reside on separate disks permits overlap of the SORT read/write cycle. You may also encounter cases where insufficient space exists on the SYS$SCRATCH disk device (for example, while Oracle Rdb builds indexes for a very large table). Using the "SORTWORK0" through "SORTWORKZ" logical names can help you avoid this problem.

Note that SORT 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 36 sort work files, it is possible to exceed the capacity of the first SORT file device and the sort operation fails never having accessed the remaining 35 sort work files.

At this time, more than 10 sort work files will only be used by the Oracle Rdb sort interface as used by the CREATE INDEX, ALTER INDEX and the clauses UNION DISTINCT, ORDER BY, GROUP BY and SELECT DISTINCT. The RMU and SQL IMPORT interfaces use the OpenVMS SORT interface which does not currently support more than 10 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 allocation within Oracle Rdb.

10.1.22 Control of Sort Work Memory Allocation

Oracle Rdb uses a built−in SORT32 package to perform many sort operations. Sometimes, these sorts exhibit a significant performance problem when initializing work memory to be used for the sort. This behavior can be experienced, for example, when a very large sort cardinality is estimated, but the actual sort cardinality is small.

In rare cases, it may be desirable to artificially limit the sort package's use of work memory. Two logicals have been created to allow this control. In general, there should be no need to use either of these logicals and misuse of them can significantly impact sort performance. Oracle recommends that these logicals be used carefully and sparingly.

The logical names are:

Table 10−1 Sort Memory Logicals

Logical	Definition
RDMS$BIND_SORT_MEMORY_WS_FACTOR	Specifies a percentage of the process's working set limit to be used when allocating sort memory for the built−in SORT32 package. If not defined, the default value is 75 (representing 75%), the maximum value is 75 (representing 75%), and the minimum value is 2 (representing 2%). Processes with vary large working set limits can sometimes experience significant page faulting and CPU consumption while initializing sort memory. This logical name can restrict the sort work memory to a percentage of the processes maximum working set.
RDMS$BIND_SORT_MEMORY_MAX_BYTES	Specifies an absolute limit to be used when allocating sort memory for the built−in SORT32 package. If not defined, the default value is unlimited (up to 1GB), the maximum value is 2147483647 and the minimum value is 32768.

10.1.23 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 columns 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 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 Example 2−2 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 Example 2−1 and Example 2−2.

The following example shows that the EMP_LAST_NAME index is used for retrieval. Any update performed will possibly be subject to the Halloween problem.

```
SQL> set flags 'strategy';
SQL> declare emp  cursor for
cont> select * from employees where last_name >= 'M' 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;
```

The following example 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.

```
SQL> set flags 'strategy';
SQL> declare emp2 cursor for
cont> select * from employees where last_name >= 'M'
cont> order by last_name 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;
```

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 cannot perform the normal protection against the Halloween

problem.

10.2 SQL Known Problems and Restrictions

This section describes known problems and restrictions for the SQL interface.

10.2.1 SET FLAGS CRONO_FLAG Removed

The SET FLAGS statement and RDMS$SET_FLAGS logical name no longer accept the obsolete keyword CRONO_FLAG. This keyword has been removed. Please update all scripts and applications to use the keyword CHRONO_FLAG.

10.2.2 Interchange File (RBR) Created by Oracle Rdb Release 7.2 Not Compatible With Previous Releases

To support the large number of new database attributes and objects, the protocol used by SQL EXPORT and SQL IMPORT has been enhanced to support more protocol types. Therefore, this format of the Oracle Rdb release 7.2 interchange files can no longer be read by older versions of Oracle Rdb.

Oracle Rdb continues to provide upward compatibility for interchange files generated by older versions.

Oracle Rdb has never supported backward compatibility, however, it was sometimes possible to use an interchange file with an older version of IMPORT. However, this protocol change will no longer permit this usage.

10.2.3 Single Statement LOCK TABLE is Not Supported for SQL Module Language and SQL Precompiler

The new LOCK TABLE statement is not currently supported as a single statement within the module language or embedded SQL language compiler.

Instead you must enclose the statement in a compound statement. That is, use BEGIN... END around the statement as shown in the following example. This format provides all the syntax and flexibility of LOCK TABLE.

This restriction does not apply to interactive or dynamic SQL.

The following extract from the module language listing file shows the reported error if you use LOCK TABLE as a single statement procedure. The other procedure in the same module is acceptable because it uses a compound statement that contains the LOCK TABLE statement.

```
1 MODULE sample_test
2 LANGUAGE C
3 PARAMETER COLONS
4
5 DECLARE ALIAS FILENAME 'mf_personnel'
6
7 PROCEDURE a (SQLCODE);
8 LOCK TABLE employees FOR EXCLUSIVE WRITE MODE;
%SQL-F-WISH_LIST, (1) Feature not yet implemented - LOCK TABLE requires compound
statement
```

```
9
10 PROCEDURE b (SQLCODE);
11 BEGIN
12 LOCK TABLE employees FOR EXCLUSIVE WRITE MODE;
13 END;
```

To workaround this problem of using LOCK TABLE for SQL module language or embedded SQL application, use a compound statement in an EXEC SQL statement.

10.2.4 Multistatement or Stored Procedures May Cause Hangs

Long−running multistatement or stored procedures can cause other users in the database to hang if the procedures obtain resources needed by those other users. Some resources obtained by the execution of a multistatement or stored procedure are not released until the multistatement or stored procedure finishes. Thus, any−long running multistatement or stored procedure can cause other processes to hang. This problem can be encountered even if the statement contains SQL 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 hangs forever.

```
Session 1:

SQL> attach 'filename MF_PERSONNEL';
SQL> create function LIB$WAIT (in real by reference)
cont> returns integer;
cont> external name LIB$WAIT location 'SYS$SHARE:LIBRTL.EXE'
cont> language general general parameter style variant;
SQL> commit;
        .
        .
        .
$ SQL
SQL> attach 'filename 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 = LIBWAIT (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, you can see that the backup process is waiting for a lock
held in the first session:

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

```
Resource: nowait signal

ProcessID Process Name        Lock ID  System ID Requested Granted
--------- ---------------     -------- -------- --------- -------
20204383  RMU BACKUP.....     5600A476  00010001 CW        NL
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 are released.

10.2.5 Use of Oracle Rdb from Shareable Images

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

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

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

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

10.3 Oracle RMU Known Problems and Restrictions

This section describes known problems and restrictions for the RMU interface.

10.3.1 RMU Convert Fails When Maximum Relation ID is Exceeded

If, when relation IDs are assigned to new system tables during an RMU Convert to a V7.2 database, the maximum relation ID of 8192 allowed by Oracle Rdb is exceeded, the fatal error %RMU−F−RELMAXIDBAD is displayed and the database is rolled back to the prior database version. Contact your Oracle support representative if you get this error. Note that when the database is rolled back, the fatal error %RMU−F−CVTROLSUC is displayed to indicate that the rollback was successful but caused by the detection of a fatal error and not requested by the user.

This condition only occurs if there are an extremely large number of tables defined in the database or if a large number of tables were defined but have subsequently been deleted.

The following example shows both the %RMU−F−RELMAXIDBAD error message if the allowed database relation ID maximum of 8192 is exceeded and the %RMU−F−CVTROLSUC error message when the database has been rolled back to V7.0 since it cannot be converted to V7.2:

```
$rmu/convert mf_personnel
%RMU−I−RMUTXT_000, Executing RMU for Oracle Rdb V7.2
Are you satisfied with your backup of
 DEVICE:[DIRECTORY]MF_PERSONNEL.RDB;1 and your backup of
 any associated .aij files [N]? Y
 %RMU−I−LOGCONVRT, database root converted to current structure level
 %RMU−F−RELMAXIDBAD, ROLLING BACK CONVERSION − Relation ID exceeds maximum
  8192 for system table RDB$RELATIONS
 %RMU−F−CVTROLSUC, CONVERT rolled−back for
  DEVICE:[DIRECTORY]MF_PERSONNEL.RDB;1 to version V7.0
```

The following example shows the normal case when the maximum allowed relation ID is not exceeded:

```
$rmu/convert mf_personnel
%RMU−I−RMUTXT_000, Executing RMU for Oracle Rdb V7.2
Are you satisfied with your backup of
 DEVICE:[DIRECTORY]MF_PERSONNEL.RDB;1 and your backup of
 any associated .aij files [N]? Y
%RMU−I−LOGCONVRT, database root converted to current structure level
%RMU−S−CVTDBSUC, database DEVICE:[DIRECTORY]MF_PERSONNEL.RDB;1
 successfully converted from version V7.0 to V7.2
%RMU−I−CVTCOMSUC, CONVERT committed for
 DEVICE:[DIRECTORY]MF_PERSONNEL.RDB;1 to version V7.2
```

10.3.2 RMU Unload /After_Journal Requires Accurate AIP Logical Area Information

The RMU Unload /After_Journal command uses the on−disk area inventory pages (AIPs) to determine the appropriate type of each logical area when reconstructing logical dbkeys for records stored in mixed−format storage areas. However, the logical area type information in the AIP is generally unknown for logical areas created prior to Oracle Rdb release 7.0.1. If the RMU Unload /After_Journal command cannot determine the logical area type for one or more AIP entries, a warning message is displayed for each such area and may ultimately return logical dbkeys with a 0 (zero) area number for records stored in mixed−format storage areas.

In order to update the on−disk logical area type in the AIP, the RMU Repair utility must be used. The INITIALIZE=LAREA_PARAMETERS=optionfile qualifier option file can be used with the TYPE qualifier. For example, to repair the EMPLOYEES table of the MF_PERSONNEL database, you would create an options file that contains the following line:

EMPLOYEES /TYPE=TABLE

For partitioned logical areas, the AREA=name qualifier can be used to identify the specific storage areas that are to be updated. For example, to repair the EMPLOYEES table of the MF_PERSONNEL database for the EMPID_OVER storage area only, you would create an options file that contains the following line:

EMPLOYEES /AREA=EMPID_OVER /TYPE=TABLE

The TYPE qualifier specifies the type of a logical area. The following keywords are allowed:

- TABLE
  Specifies that the logical area is a data table. This would be a table created using the SQL CREATE TABLE syntax.
- B−TREE
  Specifies that the logical area is a B−tree index. This would be an index created using the SQL CREATE INDEX TYPE IS SORTED syntax.
- HASH
  Specifies that the logical area is a hash index. This would be an index created using the SQL CREATE INDEX TYPE IS HASHED syntax.
- SYSTEM
  Specifies that the logical area is a system record that is used to identify hash buckets. Users cannot explicitly create these types of logical areas.

  Note

  This type should NOT be used for the RDB$SYSTEM logical areas. This type does NOT identify system relations.

- BLOB
  Specifies that the logical area is a BLOB repository.

There is no explicit error checking of the type specified for a logical area. However, an incorrect type may cause the RMU Unload /After_Journal command to be unable to correctly return valid, logical dbkeys.

10.3.3 Do Not Use HYPERSORT with RMU Optimize After_Journal Command

The OpenVMS Alpha V7.1 operating system introduced the high−performance Sort/Merge utility (also known as HYPERSORT). This utility takes advantage of the OpenVMS 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.

Because of this, 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.

10.3.4 Changes in EXCLUDE and INCLUDE Qualifiers for RMU Backup

The RMU Backup command no longer accepts both the Include and Exclude qualifiers in the same command. This change removes the confusion over exactly what gets backed up when Include and Exclude are specified on the same line, but does not diminish the capabilities of the RMU Backup command.

To explicitly exclude some storage areas from a backup, use the Exclude qualifier to name the storage areas to be excluded. This causes all storage areas to be backed up except for those named by the Exclude qualifier.

Similarly, the Include qualifier causes only those storage areas named by the qualifier to be backed up. Any storage area not named by the Include qualifier is not backed up. The Noread_only and Noworm qualifiers continue to cause read−only storage areas and WORM storage areas to be omitted from the backup even if these areas are explicitly listed by the Include qualifier.

Another related change is in the behavior of EXCLUDE=*. In previous versions, EXCLUDE=* caused all storage areas to be backed up. Beginning with V7.1, EXCLUDE=* causes only a root backup to be done. A backup created by using EXCLUDE=* can be used only by the RMU Restore Only_Root command.

10.3.5 RMU Backup Operations Should Use Only One Type of Tape Drive

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

Oracle RMU attempts to prevent using different tape drive densities during a backup operation, but is not able to detect all invalid cases and expects that all tape drives for a backup are of the same type.

As long as all of the tapes used during a backup operation can be read by the same type of tape drive during a restore operation, the backup is likely valid. This may be the case, for example, when using a TA90 and a TA90E.

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

Consult the Oracle Rdb7 Guide to Database Maintenance, the Oracle Rdb7 Guide to Database Design and Definition, and the Oracle RMU Reference Manual for additional information about Oracle Rdb backup and restore operations.

10.3.6 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 Oracle Rdb. When those cases are discovered, Oracle Rdb is corrected to prevent the introduction of the inconsistencies. It is also possible for these errors to be introduced during the normal operation of Oracle Rdb. 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 roll back 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, then the introduction of these errors is considered to be part of the normal operation of Oracle Rdb. 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.

10.4 Known Problems and Restrictions in All Interfaces for Release 7.0 and Earlier

The following problems and restrictions from release 7.0 and earlier still exist.

10.4.1 Converting Single−File Databases

Because of a substantial increase in the database root file information for V7.0, you should ensure that you have adequate disk space before you use the RMU Convert command with single−file databases and V7.0 or higher.

The size of the database root file of any given database increases a maximum of about 600 disk blocks. The actual increase depends mostly on the maximum number of users specified for the database.

10.4.2 Row Caches and Exclusive Access

If a table has a row−level cache defined for it, the Row Cache Server (RCS) may acquire a shared lock on the table and prevent any other user from acquiring a Protective or Exclusive lock on that table.

10.4.3 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:

- Reserve the table for SHARED WRITE
- Close the database and disable row cache for the duration of the exclusive transaction
- 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.

10.4.4 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 Rdb 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.

10.4.5 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 less than the smallest existing page size and the database has been manually opened or users are active, the add operation fails with the following errors:

```
%RDMS−F−NOEUACCESS, unable to acquire exclusive access to database
```

or

```
%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, any recovery scenario fails. 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.

10.4.6 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 is not 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.

10.4.7 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 catch up to the application and are not able to process database pages that are logically ahead of the application in the RDB$CHANGES system relation. 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$TRANSFERS system relation and then tries to delete any RDB$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 V4.0 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.

10.5 SQL Known Problems and Restrictions for Oracle Rdb Release 7.0 and Earlier

The following problems and restrictions from Oracle Rdb Release 7.0 and earlier still exist.

10.5.1 ARITH_EXCEPT or Incorrect Results Using LIKE IGNORE CASE

When you use LIKE...IGNORE CASE, programs linked under Oracle Rdb V4.2 and V5.1, 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 (V6.0 or higher.)

10.5.2 Different Methods of Limiting Returned Rows from Queries

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

If you define the RDMS$BIND_QG_REC_LIMIT logical name to a small value, the query often fails with no rows returned regardless of the value assigned to the logical. 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 relations 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 relation) is sufficient to read each column definition.

To see an indication of the queries executed against the system relations, define the RDMS$DEBUG_FLAGS logical name 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. 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 relations as part of query processing.

10.5.3 Suggestions for Optimal Use of 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 relation 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$BIND_BUFFERS logical name or the NUMBER OF BUFFERS IS clause in SQL to change the number of buffers).
- To distribute the disk I/O load, store the storage areas for the indexes on separate disk drives. Note that using the same storage area for multiple indexes results in contention during the index creation (Step 5) for SPAM pages.
- Consider placing the .ruj file for each parallel definer on its own disk or an infrequently used disk.
- Even though snapshot I/O should be minimal, consider disabling snapshots during parallel index creation.
- Refer to the Oracle Rdb7 Guide to Database Performance and Tuning to determine the appropriate working set values for each process to minimize excessive paging activity. In particular, avoid using working set parameters where the difference between WSQUOTA and WSEXTENT is large. The SORT utility uses the difference between these two values to allocate scratch virtual memory. A large difference (that is, the requested virtual memory grossly exceeds the available physical memory) may lead to excessive page faulting.
- The performance benefits of using SHARED DATA DEFINITION can best be observed when creating many indexes in parallel. The benefit is in the average elapsed time, not in CPU or I/O usage. For example, when two indexes are created in parallel using the SHARED DATA DEFINITION clause, the database must be attached twice, and the two attaches each use separate system resources.
- Using the SHARED DATA DEFINITION clause on a single−file database or for indexes defined in the RDB$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 ten parallel process index creations (Index1, Index2, ... Index10) and one process with ten 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 ten 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 ten of the indexes serially.

Table 10−2 Elapsed Time for Index Creations

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
All10	00:03:26.66

10.5.4 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>     language 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 Rdb.

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

10.5.6 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.

| Contents