Oracle Rdb7™ for OpenVMS

Release Notes

Release 7.0.1.3

August 1998

ORACLE®

Oracle Rdb7 Release Notes

Release 7.0.1.3

This document was prepared using VAX DOCUMENT Version 2.1.

Contents

3 Software Errors Fixed in Oracle Rdb7 Release 7.0.1.2

4 Software Errors Fixed in Oracle Rdb7 Release 7.0.1.3

5 Documentation Corrections

6 Known Problems and Restrictions

7 Enhancements in Oracle Rdb7 Release 7.0.1.1

8 Enhancements in Oracle Rdb7 Release 7.0.1.2

9 Enhancements in Oracle Rdb7 Release 7.0.1.3

Examples

Figures

Tables

Preface

Purpose of This Manual

This manual contains release notes for Oracle Rdb7 Release 7.0.1.3. The notes describe changed and enhanced features; upgrade and compatibility information; new and existing software problems and restrictions; and software and documentation corrections. These release notes cover both Oracle Rdb7 for OpenVMS Alpha and Oracle Rdb7 for OpenVMS VAX, which are referred to by their abbreviated name, Oracle Rdb7.

Intended Audience

This manual is intended for use by all Oracle Rdb7 users. Read this manual before you install, upgrade, or use Oracle Rdb7 Release 7.0.1.3.

Document Structure

This manual consists of nine chapters:

Chapter 1	Describes how to install Oracle Rdb7 Release 7.0.1.3.
Chapter 2	Describes software errors corrected in Oracle Rdb7 Release 7.0.1.1.
Chapter 3	Describes software errors corrected in Oracle Rdb7 Release 7.0.1.2.
Chapter 4	Describes software errors corrected in Oracle Rdb7 Release 7.0.1.3.
Chapter 5	Provides information not currently available in the Oracle Rdb7 documentation set.
Chapter 6	Describes problems, restrictions, and workarounds known to exist in Oracle Rdb7 Release 7.0.1.3 and CDD/Repository.
Chapter 7	Describes new features added in Oracle Rdb7 Release 7.0.1.1.
Chapter 8	Describes new features added in Oracle Rdb7 Release 7.0.1.2.
Chapter 9	Describes new features added in Oracle Rdb7 Release 7.0.1.3.

1
Installing Oracle Rdb7 Version 7.0.1.3

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

1.1 Requirements

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

- Oracle Rdb7 Release 7.0.1 must be installed on the target system.
- Oracle Rdb7 must be shutdown before you install this update kit. That is, the command file SYS$STARTUP:RMONSTOP(70).COM should be executed before proceeding with this installation. If you have an OpenVMS cluster, you must shutdown all versions of Oracle Rdb7 on all nodes in the cluster before proceeding.
- The installation requires approximately 200,000 free blocks on your system disk for OpenVMS VAX systems; 370,000 blocks for OpenVMS Alpha systems.

1.2 Invoking VMSINSTAL

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

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

variant-name

The variant names for the software update for Oracle Rdb7 Release 7.0.1.3 are:

- RDBSE3A070 for Oracle Rdb7 for OpenVMS VAX standard version.
- RDBASE3A070 for Oracle Rdb7 for OpenVMS Alpha standard version.
- RDBMVE3A070 for Oracle Rdb7 for OpenVMS VAX multiversion.
- RDBAMVE3A070 for Oracle Rdb7 for OpenVMS Alpha multiversion.

device-name

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

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

  ```
  DKA400:[RDBSE3A070.KIT]
  ```

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

  ```
  MTA0:
  ```

OPTIONS N

This parameter prints the release notes.

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

```
$ @SYS$UPDATE:VMSINSTAL RDBSE3A070 MTA0: OPTIONS N
```

1.3 Stopping the Installation

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

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

1.4 After Installing Oracle Rdb7

This update provides a new Oracle Rdb7 Oracle TRACE facility definition. Any Oracle TRACE selections that reference Oracle Rdb7 will need to be redefined to reflect the new facility version number for the updated Oracle Rdb7 facility definition, "RDBVMSV7.0-13".

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

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

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

   ```
   $ LIBRARY /TEXT /EXTRACT=RDBVMSV7.0-13 -
   _$ /OUT=RDBVMS.EPC$DEF SYS$SHARE:EPC$FACILITY.TLB
   ```

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

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

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

1.5 New Documentation HTML Save Sets Available

Included with this release is a new backup save set (RDB_702_HTML.BCK) and a new self-extracting archive file (RDB_702_HTML.EXE) for Windows NT and Windows 95. These new files contain the Oracle Rdb V7.0 (and related products) documentation in HTML format. Documentation is included for the following products:

- Oracle Rdb, Release 7.0
- Rdb Web Agent, Release 2.2
- SQL*Net for Rdb7, Release 7.1.2
- Hot Standby for Oracle Rdb and CODASYL DBMS, Release 7.0

- Distributed Option for Rdb, Release 7.0

When you expand the RDB_702_HTML.BCK backup save set, the WWW and DOC sub-directories are created and product-specific sub-directories are created below DOC. Be sure to maintain the directory structure by specifying the following command:

```
$ BACKUP RDB_702_HTML.BCK/SAVE <disk>:[directory...]
```

To access this library of documentation, point to LIBRARY.HTML using your favorite web browser.

When you expand the RDB_702_HTML.EXE self-extracting archive file for your Windows NT or Windows 95 system, the rdbhtmldocs directory is created with product-specific directories below that. Again, access this library of documentation by pointing to LIBRARY.HTML from your favorite web browser.

The PostScript format of this documentation is available in the RDB7PS.BCK backup save set.

2

Software Errors Fixed in Oracle Rdb7 Release 7.0.1.1

This chapter describes software errors that were fixed by Oracle Rdb7 Release 7.0.1.1.

2.1 Software Errors Fixed That Apply to All Interfaces

2.1.1 Problems Corrected for Strict Partitioning

Bug 546053.

When a table's storage map has the attribute PARTITIONING IS NOT UPDATABLE then mapping of data to a storage area is strictly enforced. This is known as **strict partitioning**. Oracle Rdb7 Release 7.0.1.1 corrects a problem with the strict partitioning functionality.

If the storage map was partitioned by more than one column and not all of those columns were present in the query, then the missing columns were set to LOW values when calculating the HIGH partition. This caused Rdb to scan fewer partitions than were needed to solve the query.

Consider this example:

```
CREATE TABLE STRICT_T (INFO_F INTEGER, YEAR_F INTEGER, MONTH_F INTEGER);
CREATE STORAGE MAP STRICT_M
    FOR STRICT_T
    PARTITIONING IS NOT UPDATABLE
    STORE USING (YEAR_F,MONTH_F)
        IN STRICT_1 WITH LIMIT OF (1996,12)
        IN STRICT_2 WITH LIMIT OF (1997,1)
        IN STRICT_3 WITH LIMIT OF (1997,2)
        IN STRICT_4 WITH LIMIT OF (1997,3)
        IN STRICT_5 WITH LIMIT OF (1997,4)
        IN STRICT_6 WITH LIMIT OF (1997,5)
        IN STRICT_7 WITH LIMIT OF (1997,6)
        IN STRICT_8 WITH LIMIT OF (1997,7)
        IN STRICT_9 WITH LIMIT OF (1997,8)
        IN STRICT_10 WITH LIMIT OF (1997,9)
        IN STRICT_11 WITH LIMIT OF (1997,10)
        IN STRICT_12 WITH LIMIT OF (1997,11)
        IN STRICT_13 WITH LIMIT OF (1997,12)
        OTHERWISE IN STRICT_14;
```

A query such as the following should have scanned the partitions STRICT_2 through STRICT_14 to return all rows for 1997 but the query only accessed STRICT_2 and so returned an incorrect number of rows. This was because the partitioning column MONTH_F was not referenced by the query and was not processed correctly.

```
SQL> SELECT INFO_F FROM STRICT_T WHERE YEAR = 1997;
```

This problem may be avoided by changing the query to select all partitioning columns. For example, include MONTH_F in the select list in this example. All rows are then returned because all partitioning columns are referenced and processed correctly.

This problem was corrected in Oracle Rdb7 Release 7.0.1.1. The missing columns are assumed to be NULL so that the correct data is returned.

Note

If any of the partitioning columns is assigned NULL when a row is inserted then those rows will, by default, be written to the final OTHERWISE partition. Therefore, if partitioning columns are missing from the WHERE clause then the OTHERWISE partition must also be scanned to find matches for all columns.

A more efficient strategy can be generated by enumerating all the known values in the WHERE clause. For example,

```
SQL> SELECT INFO_F FROM STRICT_T
cont>   WHERE YEAR = 1997
cont>     AND MONTH IN (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12);
```

For this example, this type of query avoids the scan of the OTHERWISE partition, and may be important if there are many partitions in the storage map. A future release of Oracle Rdb will be enhanced to produce this minimized strategy automatically for range retrievals (such as the BETWEEN operator).

2.1.2 A Bugcheck Error with Exception at RDMS$$GEN_EXPR when Using LIKE Predicate

Bugs 547400 and 545574.

In Oracle Rdb7 Release 7.0.1, produced a bugcheck error with exception at RDMS$$GEN_EXPR when using the LIKE predicate.

The following example displays a query with a LIKE predicate that may result in a bugcheck error.

```
SELECT * FROM EMPLOYEES WHERE LAST_NAME LIKE '%RAN';
```

The workaround to this problem is to specify the LIKE predicate with the IGNORE CASE clause

```
SELECT * FROM EMPLOYEES WHERE LAST_NAME LIKE '%RAN' IGNORE CASE;
```

Another workaround is to use the CONTAINING predicate instead of the LIKE predicate.

```
SELECT * FROM EMPLOYEES WHERE LAST_NAME CONTAINING 'RAN';
```

This problem was corrected in Oracle Rdb7 Release 7.0.1.1.

2.1.3 A Query with Range List Returned Wrong Result with Dynamic Optimization Disabled

Bug 520200.

The following query with a range list failed to return 1 row as expected when the dynamic optimizer was disabled by defining RDMS$MAX_STABILITY as YES:

```
    SELECT * FROM VIEW0
    WHERE
 COMPANY_ID = 'BTNYC'
 AND LEGAL_ENTITY_ID IN ('NYO', '*' )
 AND CPTY_ID = 'FFTW N  ADI' ;
```

This query used the following strategy:

```
Cross block of 2 entries
  Cross block entry 1
    Conjunct        Get     Retrieval by index of relation CONFO_OUT_GEN
      Index name  CONFO_OUT_GEN_IDX [1:1] Bool
  Cross block entry 2
    Conjunct        Aggregate       Conjunct        Get
    Retrieval by index of relation CONFO_OUT_GEN
      Index name  CONFO_OUT_GEN_IDX [4:4] Bool
```

Even though the dynamic optimizer was disabled, the new cost model in Oracle Rdb7 caused the optimizer to choose the joined index key (COMPANY_ID) to join the 2 tables and then use "CONFO_OUT_GEN_IDX [4:4] Bool" to retrieve the index. The optimizer then unnecessarily processed the range list retrieval blocks and chose the best one from the latter of the IN predicates "LEGAL_ENTITY_ID IN ('NYO', '*')". Thus, the wrong index key segment was generated for index retrieval.

A workaround to this problem is to enable the dynamic optimizer by deassigning the logical name RDMS$MAX_STABILITY, and the query will use the dynamic strategy with background indices.

This problem was fixed in Oracle Rdb7 Release 7.0.1.1.

2.1.4 EXCESS_TRAN Error when Using 2PC Transactions

Bug 546833.

When a program made multiple attaches where at least one of the attaches was remote, it was possible to receive an RDB-F-EXCESS_TRANS, exceeded limit of 1 transaction per database. This would happen after a distributed (DECDTM) transaction involving all the databases failed to start due to a condition such as a lock conflict. That is, transactions would start on some of the databases, but not on all of them.

When the partially started transaction was then aborted, DECDTM was not being properly informed and did not communicate the rollback to the remote databases. The next time a transaction was started involving one of these remote databases, that database would report that a transaction was already active.

There is no workaround to this problem other than aborting the job to clear the transaction on the remote database.

This problem was corrected in Oracle Rdb7 Release 7.0.1.1.

2.1.5 Excessive Snapshot File Growth when the Number of Cluster Nodes Set to 1

Bug 545595.

A change was made to the snapshot garbage collection algorithm in Oracle Rdb7 which could result in snapshot files growing excessively and extending unnecessarily. For the problem to occur, it was necessary that the NUMBER OF CLUSTER NODES value for the database was set to 1 and that users were attaching to the database, updating and then detaching (rather than staying attached for long periods).

In order to find a suitable snapshot page to use, a read/write transaction takes out a special lock for the snapshot file in question. This is called the snap area cursor (SAC) lock and contains, within the lock value block, the page within the snapshot file where space was last found. Prior to Oracle Rdb7 the monitor process owned these locks so that they were maintained continuously while the database was open. In Oracle Rdb7 an optimization was made to reduce the number of locks the monitor owned, in the case that NUMBER OF CLUSTER NODES was set to 1. In this case the monitor would not own the SAC locks, rather they would be taken out by the individual users when they touched a storage area and retained by that user until they detached from the database.

In the case of a user attaching to the database, making some updates or inserts and then detaching, it was possible that the value of the lock value block could be lost and therefore lose the location to start searching for a snapshot page. Then the search would begin from the beginning of the area and, if the pages at the beginning of the area could not be garbage collected, the snapshot file would be extended, regardless of what space might be available in the rest of the snapshot file.

The workaround to this problem is to alter the database so that the number of cluster nodes is set to more than 1. This is an offline activity.

This problem was corrected in Oracle Rdb7 Release 7.0.1.1.

2.1.6 Syntax Error not Generated when OTHERWISE Clause Used Incorrectly

When creating a vertical record partition storage map, a syntax error should be generated if the OTHERWISE clause is used incorrectly. A vertical record partition storage map requires the use of STORE COLUMNS syntax and also allows the use of an optional STORE clause. The syntax does not allow a catchall OTHERWISE clause as a way of denoting a partition.

In the example below, a syntax error is generated because the OTHERWISE clause is used incorrectly when creating the vertical record partition storage map VRP_MAP.

```
ATTACH 'FILE MF_PERSONNEL';
-- create a table
CREATE TABLE VRP_TABLE (COL1 INT, COL2 INT, COL3 INT, COL4 INT);
commit;
-- This should fail
  CREATE STORAGE MAP VRP_MAP FOR VRP_TABLE
  STORE COLUMNS (COL1,COL3) IN EMPIDS_LOW
  STORE COLUMNS (COL4) IN EMPIDS_MID
  OTHERWISE IN EMPIDS_OVER;
  OTHERWISE IN EMPIDS_OVER;
  ^
%SQL-W-LOOK_FOR_STT, Syntax error, looking for:
%SQL-W-LOOK_FOR_CON,   (, STORE, ENABLE, DISABLE, PLACEMENT,
%SQL-W-LOOK_FOR_CON,   THRESHOLD, THRESHOLDS, REORGANIZE,
%SQL-W-LOOK_FOR_CON,   PARTITIONING, ;,
%SQL-F-LOOK_FOR_FIN,  found OTHERWISE instead
```

The following example shows the correct syntax for creating a Vertical Record Partition Storage Map:

```
ATTACH 'FILE MF_PERSONNEL';
-- this should succeed
CREATE STORAGE MAP VRP_MAP FOR VRP_TABLE
  STORE COLUMNS (COL1,COL3) IN EMPIDS_LOW
  STORE COLUMNS (COL4) IN EMPIDS_MID
  STORE IN EMPIDS_OVER;
ROLLBACK;
```

This problem was corrected in Oracle Rdb7 Release 7.0.1.1.

2.1.7 RMU/VERIFY BADNODEID Error with Sorted Ranked Indexes

Using sorted ranked indexes with a very large number of duplicates, overflow duplicate chains could be created with invalid index identifications.

The RMU/VERIFY command detects this problem as in the following example:

```
  $ RMU/VERIFY/ALL DUA0:[DB]DB.RDB
  %RMU-E-BADNODEID, index id invalid for b-tree node 84:755:1
                    expected: 0054 (hex), found 002E (hex)
  %RMU-I-DUPOWNDBK, Dbkey of owner of this duplicate node is 84:754:0
  %RMU-I-BTRDUPCAR, Inconsistent duplicate cardinality (C1) of 714
                    specified for entry 1 at dbkey 84:754:0.
                    Actual count of duplicates is 266.
  %RMU-I-BTRROODBK, root dbkey of B-tree is 84:754:0
```

The index ID of the index node is incorrect. When the index node was created, an invalid index ID was used.

To work around this problem, drop and re-create the index.

This problem was corrected in Oracle Rdb7 Release 7.0.1.1. The overflow duplicate index node now is created with the correct index ID.

2.1.8 Possible Data Corruption Using ALTER TABLE ... ADD COLUMN with DEFAULT

Bug 548608.

When a table is altered to add new columns, change datatypes, or remove obsolete columns Oracle Rdb keeps track of the changed versions of the system metadata in a table called RDB$FIELD_VERSIONS. By keeping track of the old metadata it is possible for ALTER TABLE to quickly change the table structure without accessing and updating the rows in the target table.

Some operations such as the following would attempt to purge the old version rows when it was known that all the rows in the table were at the same version.

- ALTER TABLE ... ADD COLUMN with DEFAULT

  When a new column is added with a DEFAULT, the SQL92 Database Language standard dictates that all rows which existed at the time of the ALTER TABLE will inherit the default for the column. This operation requires an UPDATE of all rows in the table which will bring the table to the current version of the row.

- TRUNCATE TABLE (Oracle Rdb7 and later versions)

  When TRUNCATE TABLE completes successfully, no rows remain in the table.

- ALTER TABLE ... ALTER COLUMN data type when RDMS$BIND_UPDATE_CHANGED_RELATION is defined.

  This logical name forces all rows to be updated to the latest version when a column's datatype is modified.

A problem was reported which involved performing one of these commands when some other process had an old version of the table metadata loaded. Oracle Rdb incorrectly purged the old version metadata which was in use by the other process. The result was that rows subsequently stored by that process could not be later decoded correctly and could unexpectedly return NULLs as column values.

Examination of these rows using RMU/DUMP/AREA showed that the stored version number was no longer registered in the RDB$FIELD_VERSIONS system table. The workaround is to delete and re-insert the affected data. If this is not possible or practical then customers should contact Oracle World Wide Support for assistance in rebuilding the missing RDB$FIELD_VERSIONS data. Typically, this can be done if you have documentation on the table modifications or a backup of an older version of the database exists.

In Oracle Rdb7 Release 7.0.1.1, these operations no longer attempt to purge the system table metadata unless you have attached to the database using the RESTRICTED ACCESS clause as in the following example.

```
SQL> ATTACH 'FILENAME yourdatabase RESTRICTED ACCESS';
```

This guarantees that no other process has cached a stale version of the table metadata.

2.1.9 Recursive Logical Name Caused Database Attach to Loop

Bug 401369.

If a logical name was recursively or incorrectly defined, Oracle Rdb7 could loop during a database attach operation.

In the following example, Oracle Rdb7 would not return from the database attach and the process would be stuck in an infinite loop:

```
$ DEFINE ADB BDB
$ DEFINE BDB CDB
$ DEFINE CDB ADB
$ MCR SQL$
SQL> ATTACH 'FILE ADB';
```

Use the DCL command SHOW LOGICAL to verify the current result of translation on the specified logical name(s).

This problem was corrected in Oracle Rdb7 Release 7.0.1.1. A limit of 32 levels of logical name translation is now enforced by Oracle Rdb7. On OpenVMS systems, RMS may further limit a logical name translation depth to 10 levels.

The following example, demonstrates the correct action:

```
$ DEFINE ADB BDB
$ DEFINE BDB CDB
$ DEFINE CDB ADB
$ MCR SQL$
SQL> ATTACH 'FILE ADB';
%SQL-F-ERRATTDEC, Error attaching to database ADB
-RDB-E-BAD_DB_FORMAT, ADB does not reference a database known to Rdb
-RMS-F-LNE, Logical name translation error
```

2.1.10 Corrected Tracing of Constraint Evaluation

In prior versions of Oracle Rdb you could define the logical name RDMS$DEBUG_FLAGS to "Sn", or use the SQL SET FLAGS 'STRATEGY, REQUEST_NAME' statement to direct Rdb to trace the execution of constraints at runtime.

If a constraint was solved by collecting a list of database keys instead of performing a scan of the table then this trace message was printed once per database key which was misleading, and would consume space in the output log file.

This problem was corrected in Oracle Rdb7 Release 7.0.1.1. These trace messages are now printed just once per evaluation.

In addition, the trace message has also been changed to better describe the type of evaluation. For example,

- Constraint "PKEY" evaluated (create)

  This message traces the evaluation of the constraint PKEY when it is added to a table during CREATE TABLE, or ALTER TABLE.

- Constraint "U_VALUES" evaluated (commit)

  This message traces the evaluation of the constraint U_VALUES at commit time, or when the SET CONSTRAINTS statement is used. It indicates that the constraint is DEFERRABLE.

- Constraint "B_UNIQUE" evaluated (statement)

  This message traces the evaluation of the constraint B_UNIQUE at statement time. It indicates that a NOT DEFERRABLE constraint is delayed until the statement end when the dialect is set to SQL92, or ORACLE LEVEL1. Typically, the SQL statement is updating multiple rows of a table and any single update would normally violate a NOT DEFERRABLE constraint. By delaying the constraint evaluation until all changes are made the constraint can be successfully checked.

- Constraint "B_UNIQUE" evaluated (verb)

  This message traces the evaluation of the constraint B_UNIQUE at verb time (statement end). It indicates that the constraint is NOT DEFERRABLE, or it had its evaluation time modified on the SET TRANSACTION statement.

- Constraint "U_VALUES" evaluated (verify)

  This message traces the evaluation of the constraint U_VALUES during verify by the RMU/VERIFY/CONSTRAINT command.

- Constraint "VV_CHECKOPT1" evaluated (view)

This message traces the evaluation of the constraint VV_CHECKOPT1 during INSERT or UPDATE into a view which has the WITH CHECK OPTION.

2.1.11 Missed Transitivity in Query Could Produce Wrong Results

Bug 531362.

A side effect of a correction introduced in Oracle Rdb7 Release 7.0.1 was that a limited class of queries could potentially return wrong results. Here is one example of such a query:

```
SELECT DISTINCT
       LOAN.BORROWER_ID, LOAN.LOAN_ABBREVIATION, LOAN.BELLWETHER,
       LNMA.LOAN_ID, LNMA.MATURITY_ID, LNMA.MATURITY_DATE, LNMA.CURRENCY_ID
FROM   BORROWER BORR,
       LOAN LOAN,
       LOAN_MATURITY LNMA
WHERE  BORR.BORROWER_ID        =1992000007 AND
       LNMA.BORROWING          = 'Y' AND
       LNMA.MATURITY_DATE      >= '12-MAY-1997 00:00:00.00' AND
       BORR.BORROWER_ID        = LOAN.BORROWER_ID AND
       LOAN.LOAN_ID            = LNMA.LOAN_ID ;
```

The problem was that the optimizer did not always recognize the transitivity between tables BORROWER and LOAN on column BORROWER_ID. For example, it did not notice that, because `BORR.BORROWER_ID = 1992000007` and `BORR.BORROWER_ID = LOAN.BORROWER_ID`, then `LOAN.BORROWER_ID = 1992000007`. As a result, it failed to implement the proper restriction on the LOAN table to only retain the rows for the selected BORROWER_ID.

Note that the problem only happened when the join between the LOAN and BORROWER tables was implemented using the match technique. In this particular example, the match technique was a poor choice, and only used by the optimizer because the cardinalities in the database had not been updated with the proper index prefix cardinalities. Once those were in place, the optimizer switched to a more efficient strategy for which the problem does not exist.

The workaround is to code the transitivity explicitly. For example, in the above case, add the following selection criteria:

```
LOAN.BORROWER_ID = 1992000007 AND  . . .
```

This problem was corrected in Oracle Rdb7 Release 7.0.1.1.

2.1.12 Bucheck Error at DIOBND$GET_LACB + 00000088

Bug 504889.

In some rare cases Oracle Rdb would generate a bugcheck error with the following exception message when fetching LIST OF BYTE VARYING data:

```
***** Exception at 004AD1C3 : DIOBND$GET_LACB + 00000088
Internal consistency failure
```

This happened when an application accessed LIST OF BYTE VARYING (SEGMENTED STRING) data in a concurrent environment. The LIST data being fetched had been changed by another user process before an attempt to read the LIST was made by the application. The bugcheck error occurred because the pointer held in the fetched row was stale and now pointed to a fragment of another LIST column. This requires that the pointer be held across a COMMIT of a transaction.

The use of stale LIST pointers can not be corrected by Oracle as this is a function of the application software. However, Oracle recommends that LIST column data be used in the same transaction from which the LIST column was fetched. This will avoid this problem in the future.

This bugcheck error no longer occurs with Oracle Rdb7 Release 7.0.1.1. Oracle Rdb now validates that the first segment of the LIST is a valid first segment. Rdb will now return the following error during the OPEN of the list cursor:

```
%RDB-E-BAD_SEGSTR_ID, invalid segmented string identifier
```

2.1.13 Excessive SPAM Fetches During Sequential Scan

Bug 471774.

During a sequential scan of a uniform format storage area, an inordinate number of SPAM page fetches occurred. For example, a sequential scan of a storage area with 10,000 pages could incur over 50,000 SPAM fetches. Of course, because the SPAM pages were frequently accessed, they tended to stay in the buffer pool and probably did not require I/Os. The excessive number of SPAM fetches did, however, consume more CPU resources.

A possible workaround to the problem of excessive SPAM fetches is to disable the asynchronous prefetch (APF) feature. Although disabling APF can reduce the CPU usage for the SPAM searches during these sequential scans, this will likely result in slower over-all performance because the wait time for the disk may be longer than the CPU resource usage for the scans.

This problem was corrected in Oracle Rdb7 Release 7.0.1.1. The number of SPAM page fetches has been reduced. SPAM pages must still be consulted in order to determine what pages of a uniform format storage area contain the specified logical area but the number of fetches should be less.

2.1.14 Database File Creation on Disks with OpenVMS File High-Water Marking Enabled

Oracle Rdb7 database file creation on disks with OpenVMS file high-water marking enabled could take much longer then on disks without file high-water marking. This was largely due to OpenVMS erasing the contents of the file prior to the file being initialized. In most cases, file high-water marking being enabled caused the database file creation operation to take about twice as long as when file high-water marking was not enabled.

A workaround for this problem is to disable file high-water marking for the duration of the database file creation. Once the creation is complete, file high-water marking can be re-enabled. The DCL command SET VOLUME is used to enable and disable the file high-water marking attribute for a disk.

This situation was improved in Oracle Rdb7 Release 7.0.1.1. Where possible, Oracle Rdb7 now uses file creation and access attributes to speed the creation and initialization of database files located on disks with file high-water marking enabled.

2.1.15 DBR Bugcheck Error at UTIO$READ_BLOCK Reading a Large RUJ File

During a database recovery rollback of a very large transaction with an RUJ file larger than 2GB, the DBR process could fail and produce a bugcheck error at UTIO$READ_BLOCK. The following example shows this information from a VAX bugcheck error file:

```
$ SEARCH RDMDBRBUG.DMP "EXCEPTION","-F-","-W-","SAVED PC"
***** Exception at 000733CB : UTIO$READ_BLOCK + 000000F3
%RDMS-F-FILACCERR, error reading disk file
-SYSTEM-W-ENDOFFILE, end of file
Saved PC = 80000014 : S0 address
Saved PC = 0002687C : DBR$GET_RUJ_FIELD + 000000C6
Saved PC = 0002679D : DBR$GET_RUJ + 00000018
Saved PC = 000263EC : DBR$RESOLVE + 00000504
Saved PC = 0002413D : DBR$RECOVER + 00000629
Saved PC = 00023938 : DBR + 00000690
```

This exception was due to the DBR process incorrectly calculating the virtual block number in the RUJ file when the RUJ file exceeded 2 billion bytes.

This problem was corrected in Oracle Rdb7 Release 7.0.1.1. The block number in the RUJ file is now correctly calculated.

2.1.16 Query on a View Containing a CASE Statement Returned the Wrong Result

Bug 476557.

The following query on a view containing a CASE statement returned the wrong results:

```
SELECT COUNTER_BY, COUNT(DISTINCT COUNTER_AT) FROM COMPS_VIEW
    GROUP BY COUNTER_BY;
 COUNTER_BY
 Jones                  2
 UK                     1
 Wiles                  1
 Wiles                  2
4 rows selected
! the above query should return the following:
!
! COUNTER_BY
! France                 2
! Jones                  1
! UK                     1
! Wiles                  2
!4 rows selected

! The comps_view view is defined as :
```

```
CREATE VIEW COMPS_VIEW (
        COUNTER_ON,
        COUNTER_AT,
        COUNTER_BY)
    AS
     SELECT
                COUNTER_ON,
            CASE
                WHEN RESPONSIBILITY = 'R'
                THEN COUNTER_AT
                ELSE AUTHORIZATION_AT
            END,
            CASE
                WHEN RESPONSIBILITY = 'R'
                THEN COUNTER_BY
                ELSE AUTHORIZATION_BY
            END
            FROM COMPLAINTS;
```

The problem was fixed in Oracle Rdb7 Release 7.0.1.1.

2.1.17 Database Hung when User Process Did Not Release Freeze Lock

Bug 512724.

It was possible for a database to become hung when a database recovery process (DBR) attempted to recover a failed user process. This particular hang would occur when a user process did not release the Oracle Rdb7 freeze lock needed by the DBR. Investigation of the user process showed that the internal Oracle Rdb7 lock data structures indicated that the process was not holding the lock even though it was, in fact, holding the lock. If the user process was deleted, by using the DCL STOP command for example, the DBR was able to proceed and the database would no longer be hung.

The problem was caused by Oracle Rdb7 not properly taking into account subtle changes in the behavior of the $ENQ system service when OpenVMS dynamic lock remastering was occurring. OpenVMS would not always provide the lock identification for the freeze lock before returning from an asynchronous $ENQ call. This created a race condition that would cause Oracle Rdb7 to occasionally lose track of the state of the freeze lock.

A workaround for this problem is to disable OpenVMS dynamic lock remastering by setting the SYSGEN parameter PE1 to a low value. This should be some value less than the number of locks used in a database, such as 40. This prevents OpenVMS from attempting to remaster the database lock tree.

This problem was corrected in Oracle Rdb7 Release 7.0.1.1.

2.1.18 Poor Performance on Dynamic Optimization Strategies

Bug 520711

The dynamic optimizer combines multiple indexes at run time to select matching rows from a table. When the optimizer notices that an index is unproductive, it discards it. One reason for doing so is if the cost of scanning the index proves more expensive than a calculated threshold. That cost is a simple count of the number of I/Os being performed while scanning the index.

The problem was that the count of I/Os only included *synchronous* I/Os. However, release 6.1 of Oracle Rdb introduced a mechanism to automatically detect quasi-sequential access patterns and predict future accesses to database pages, allowing it to pre-fetch those pages asynchronously. This is a very common situation when

scanning a range of keys from a B-tree index when the index nodes reside on adjacent pages.

Unfortunately, the dynamic optimizer was not counting those *asynchronous* I/Os and therefore would unknowingly continue scanning an unproductive index, adding greatly to the execution cost of the query.

The following example shows a trace of the dynamic execution of a query that uses a combination of 3 indexes. The first situation shows that after having obtained 79 dbkeys from the first index, the dynamic optimizer proceeds with the second then third index. The apparent cost for scanning the second index (5+10 = 15 I/Os) seems low, but that cost only includes the first few synchronous I/Os. It does not include the large number of asynchronous I/Os needed to scan the entire index (the index holds over 6 million keys). The total time needed to complete the query is over 4 minutes.

```
~E#0008.01(1) Estim   Ndx:Lev/Seps/DBKeys 1:2/3\44 2:_133050 3:5/11\381397 4:5/11\381397
~E#0008.01(1) BgrNdx1 EofData  DBKeys=79  Fetches=1+1  RecsOut=0 #Bufs=43
~E#0008.01(1) BgrNdx2 EofData  DBKeys=79  Fetches=5+10  RecsOut=0 #Bufs=43
~E#0008.01(1) BgrNdx3 EofData  DBKeys=46  Fetches=4+5  RecsOut=0 #Bufs=43
~E#0008.01(1) Fin     Buf      DBKeys=46  Fetches=0+30  RecsOut=46
```

The second situation shows the dynamic optimizer correctly discarding the second and third indexes. The total time to complete the query is now less than a second.

```
~E#0012.01(1) Estim   Ndx:Lev/Seps/DBKeys 1:2/3\44 2:_133050 3:5/11\381397 4:5/11\381397
~E#0012.01(1) BgrNdx1 EofData  DBKeys=79  Fetches=1+1  RecsOut=0 #Bufs=43
~E#0012.01(1) BgrNdx2 FtchLim  DBKeys=0  Fetches=5+21  RecsOut=0
~E#0012.01(1) BgrNdx3 FtchLim  DBKeys=0  Fetches=4+10  RecsOut=0
~E#0012.01(1) Fin     Buf      DBKeys=79  Fetches=0+43  RecsOut=49
```

Note that the problem is more likely to happen with "well organized" indexes, where the physical ordering of nodes on database pages closely matches the logical ordering of the keys.

A workaround is to disable the automatically detected aysnchronous pre-fetch mechanism, via the following commands:

```
SQL> ALTER DATABASE FILENAME ....
cont> DETECT ASYNC PREFETCH DISABLED;
```

This problem was corrected in Oracle Rdb7 Release 7.0.1.1.

2.1.19 Recovery Operation Work File Allocation

During a recovery (roll-forward) operation, Oracle Rdb7 may need to allocate work files. In previous versions of Oracle Rdb7, these work files were created without a specified allocation or extend size. This would frequently cause the files to be extended as they were being accessed. In turn, additional I/Os would occur during recovery and could slow down the recovery operation.

This problem was corrected in Oracle Rdb7 Release 7.0.1.1. Oracle Rdb now allocates recovery work files with larger allocation and extension values and attempts to "tune" the values based on the use of previous work files during the recovery operation.

As a workaround on OpenVMS, specify a larger RMS sequential file extension value with the SET RMS_DEFAULT command at the DCL prompt as in the following example:

```
$ SET RMS_DEFAULT /SEQUENTIAL /EXTEND_QUANTITY=512
```

Note

The additional file size may cause recovery operations to require slightly more disk space. Make sure to reserve enough disk space for the recovery work files.

2.1.20 Database File Creation Failure Left Partially Created Files

When database file creation failed, due to problems such as inadequate disk space, the partially created file could be left with a file size that appeared larger than the file's actual length.

The following example shows this problem:

```
SQL> ALTER DATABASE FILENAME FOO
       ADD STORAGE AREA A1 ALLOCATION 1000000;
%RDB-F-SYS_REQUEST, error from system services request
-RDMS-F-FILACCERR, error extending file DUA0:[DB]A1.RDB;
-SYSTEM-W-DEVICEFULL, device full; allocation failure
SQL> EXIT

$ DIRECTORY /NOHEADING /SIZE=ALLOCATION A1.RDB
DUA0:[DB]A1.RDB          2000008/0
```

This problem has been corrected in most cases. When file creation fails and when the file is not being created on a disk "bound volume set" no partially created files will be left on the system.

2.1.21 Some Conditional Expressions Returned Incorrect Results

Bugs 515220 and 499518.

In previous versions of Oracle Rdb, some queries that contained conditional expressions (CASE, COALESCE, DECODE, NULLIF, NVL) and referenced views could return incorrect results.

More specifically, this problem may have been seen in the following two scenarios:

- If a conditional expression referenced columns from a view that contained one or more UNION clauses.

```
SELECT * FROM TABLE1, VIEW1
WHERE TABLE1.COL = COALESCE(VIEW1.COL,0);
```

- If a view contained a conditional expression in a DISTINCT select list.

```
CREATE VIEW VIEW2 (ID, NUM)
    AS SELECT distinct ID,
        CASE
            WHEN (NUMBER < 10) THEN NUMBER
            ELSE (NUMBER + 1)
        END
    FROM TABLE1;
SELECT * FROM VIEW2 WHERE ID = '123';
```

This problem was corrected in Oracle Rdb7 Release 7.0.1.1. Rdb now properly processes conditional expressions in the described scenarios returning the correct results.

2.2 SQL Errors Fixed

2.2.1 JOURNAL IS UNSUPPRESSED Syntax Not Recognized

Bug 440536.

In all prior versions of Oracle Rdb the JOURNAL IS UNSUPPRESSED clause was documented for the ALTER DATABASE and ALTER JOURNAL commands but was rejected as illegal syntax.

```
SQL> ALTER DATABASE FILENAME db$:scratch
cont> ALTER JOURNAL RDB$JOURNAL
cont> JOURNAL IS UNSUPPRESSED;
JOURNAL IS UNSUPPRESSED;
          ^
%SQL-F-LOOK_FOR, Syntax error, looking for ENABLED, found UNSUPPRESSED instead
```

As mentioned in the error message, SQL was expecting the keyword ENABLED.

This problem was corrected in Oracle Rdb7 Release 7.0.1.1. SQL now accepts the keyword UNSUPPRESSED as documented, as well as the alternate keyword ENABLED which was accepted by prior versions.

The workaround is to use the ENABLED keyword in the the ALTER JOURNAL clause.

2.2.2 SQLMOD/CONTEXT=() with MSPs Produced CTXPARMNOTALL Error

Bug 440523.

SQL module language disallowed some statements in multistatement procedures when compiled with the /CONTEXT = (procedure_name) switch. The disallowed statements included SET TRANSACTION, COMMIT, ROLLBACK, and GET DIAGNOSTICS. A SQL-F-CTXPARMNOTALL error was produced when any of these statements were in a multistatement procedure that was listed in the procedure list of the /CONTEXT switch. SQL was trying to prevent mixing of internal and external transactions by doing checks at compiletime that should be handled by Rdb at runtime.

The SQL-F-CTXPARMNOTALL error is no longer generated in these cases. Any transaction problems that may arise from using the /CONTEXT=(procedure_ name) switch with multistatement procedures that contain any of these statements should be checked for at runtime by Rdb.

The following example shows the SQL-F-CTXPARMNOTALL error being generated when compiling a module using the SQLMOD/CONTEXT=(procedure_ name) command, where the multistatement procedure contains a SET TRANSACTION statement. Notice that the error points to the BEGIN statement for the multistatement procedure and not to the statement actually causing the problem.

```
$ CREATE FILE X.SQLMOD
MODULE X
DIALECT SQL92
LANGUAGE GENERAL

DECLARE ALIAS FILENAME 'TEST'
PROCEDURE TXN_READ_ONLY
      SQLSTATE;
BEGIN
SET TRANSACTION READ ONLY;
END;
```

```
$ SQLMOD/CONTEXT=(TXN_READ_ONLY) X.SQLMOD
BEGIN
1
%SQL-F-CTXPARMNOTALL, (1) This statement not allowed in procedure in CONTEXT list
```

This problem was corrected in Oracle Rdb7 Release 7.0.1.1.

2.2.3 SQLMOD/C_PROTOTYPES Produced a Bugcheck Error in SQL$$INSERT_CC_PARAM_DECL

Bug 526214.

SQLMOD produced a bugcheck error when it encountered a datatype that it didn't expect while generating a C prototype for a SQL module language routine. In the most recent known case, the datatype was a user declared domain based on timestamp; this did not occur using timestamp itself.

Instead of having this happen again for each datatype that was not explicitly handled, SQL will now assume a C datatype of void when it doesn't recognize the datatype of the SQL module language parameter, so the parameter in the C prototype will be void *.

The following example shows SQL module language producing a bugcheck error when encountering a parameter whose datatype is a user-defined domain based on timestamp.

```
$ SQL
SQL> ATTACH 'FILENAME TEST';
SQL> CREATE DOMAIN TS TIMESTAMP;
SQL> COMMIT;
SQL> EXIT;
```

```
$ CREATE FILE X.SQLMOD
MODULE          X
DIALECT         SQL92
LANGUAGE        C
PARAMETER       COLONS

DECLARE ALIAS FILENAME 'TEST'

PROCEDURE COPY_TIMESTAMP(
    SQLCODE,
    :ts_in       ts,
    :ts_out      ts );
BEGIN
SET :ts_out = :ts_in;
END;
```

```
$ SQLMOD/C_PROTOTYPES X.SQLMOD
%SQL-I-BUGCHKDMP, generating bugcheck dump file USER1:[work]SQLBUGCHK.DMP;
```

The following exception is in the bugcheck file.

```
***** Exception at 00433113 : SQL$$INSERT_CC_PARAM_DECL + 000001AA
%SQL-F-BUGCHK, There has been a fatal error. Please contact your
Oracle support representative. COB$GENDML - 22
```

There are no recommended workarounds. The user can either try to isolate the unknown datatype and avoid using it, or not use the /C_PROTOTYPES switch.

This problem was corrected in Oracle Rdb7 Release 7.0.1.1.

2.2.4 The DECLARE LOCAL TEMPORARY TABLE Statement Did Not Support DECIMAL/NUMERIC Datatypes

In prior releases of Oracle Rdb7 attempts to use the datatypes DECIMAL or NUMERIC for a DECLARE LOCAL TEMPORARY TABLE statement would fail if the dialect was set to SQL92.

The failure occurred in the CREATE MODULE statement:

```
SQL> SET DIALECT 'SQL92';
SQL> ATTACH 'FILE DB$:SCRATCH';
SQL>
SQL> CREATE MODULE M2 LANGUAGE SQL
cont>  DECLARE LOCAL TEMP TABLE MODULE.T (A NUMERIC(5))
cont>  PROCEDURE P2;
cont>   BEGIN
cont>   END;
cont> END MODULE;
%SQL-I-NO_NUMERIC, A is being converted from NUMERIC to INTEGER.
%RDB-E-NO_META_UPDATE, metadata update failed
-RDMS-E-BAD_CODE, corruption in the query string
```

The failure also occurred in the DECLARE LOCAL TEMPORARY TABLE statement in interactive or dynamic SQL.

```
SQL> SET DIALECT 'SQL92';
SQL> ATTACH 'FILE DB$:SCRATCH';
SQL> DECLARE LOCAL TEMP TABLE MODULE.T (A NUMERIC(5));
%SQL-I-NO_NUMERIC, A is being converted from NUMERIC to INTEGER.
%RDMS-E-BAD_CODE, corruption in the query string
```

The workaround to this problem is to leave the dialect as the default when creating the stored module or declaring the temporary table.

This problem was corrected in Oracle Rdb7 Release 7.0.1.1. These datatypes are now supported for use in declared local temporary tables. This problem did not occur for temporary tables created using the CREATE LOCAL TEMPORARY TABLE, or CREATE GLOBAL TEMPORARY TABLE statements.

2.2.5 TRUNCATE TABLE Not Allowed on Temporary Table During Read-Only Transaction

In prior releases of Oracle Rdb7, the TRUNCATE TABLE statement would fail if a read-only transaction was active. However, for a local or global temporary table this command should have succeeded, because it is equivalent to a DELETE FROM statement. The following example shows the problem:

```
SQL> CREATE GLOBAL TEMPORARY TABLE T (A INTEGER) ON COMMIT PRESERVE ROWS;
SQL> INSERT INTO T VALUES (1);
1 row inserted
SQL> COMMIT;
SQL>
SQL> SET TRANSACTION READ ONLY;
SQL> SELECT * FROM T;
           A
           1
1 row selected
SQL> TRUNCATE TABLE T;
%RDB-E-NO_META_UPDATE, metadata update failed
-RDB-E-READ_ONLY_TRANS, attempt to update during a read-only transaction
```

The workaround to this problem is to use a DELETE FROM statement against the temporary table.

This problem was corrected in Oracle Rdb7 Release 7.0.1.1. The TRUNCATE TABLE statement is now permitted on a temporary table during a read-only transaction.

2.2.6 Restriction for ATOMIC Compound Statements

When the ATOMIC keyword is used in a compound statement (BEGIN . . . END block) then all statements in the block must succeed otherwise all statements will be rolled back; this includes the failing statement and all prior statements which succeed up to the BEGIN.

This also means that no statement within that block may COMMIT or ROLLBACK a transaction. SQL prevents the COMMIT and ROLLBACK statements from appearing in a BEGIN . . . END section. However, it was possible in Oracle Rdb7 to use the new CALL statement in the compound statement and indirectly execute a COMMIT or ROLLBACK statement using the called procedure. By doing so, the ATOMIC attribute of the compound statement was violated.

This problem was corrected in Oracle Rdb7 Release 7.0.1.1. Attempts to call a stored procedure which performs a COMMIT or ROLLBACK from an ATOMIC compound statement will now result in this error:

```
%RDB-E-INV_TRANS_ACTIO, transaction state can not be modified from this call
```

This example shows a definition which now fails.

```
SQL> CREATE MODULE DEMO
cont>   LANGUAGE SQL
cont>
cont> PROCEDURE P1 (IN :EMPID CHAR(5), IN :CITY CHAR(10));
cont> BEGIN
cont>   UPDATE EMPLOYEES SET CITY = :CITY WHERE EMPLOYEE_ID = :EMPID;
cont>   COMMIT;
cont>   INSERT INTO EMPLOYEES (EMPLOYEE_ID) VALUES (:EMPID);
cont> END;
cont>
cont> END MODULE;
SQL> COMMIT;
SQL>
SQL> BEGIN ATOMIC
cont> CALL P1 ('00164', 'Paris');
cont> END;
%RDB-E-INV_TRANS_ACTIO, transaction state can not be modified from this call
SQL>
```

2.2.7 Incorrect Processing of Subquery when Nested in FOR Cursor Loop

Bugs 398992, 392543, and 468661.

A subquery could return incorrect results when it appeared in a SET statement, CASE statement, or an UPDATE ... WHERE CURRENT OF statement nested within a FOR cursor loop and this subquery references local variables or procedure parameters initialized inside the FOR cursor loop.

This problem was due to an optimization which pulled the subquery evaluation into the FOR cursor loop's own query, thereby evaluating it before the local variables or parameters had been initialized.

The following example shows the problem:

```
SQL> SET FLAGS 'TRACE';
SQL>
SQL> BEGIN
cont> DECLARE :ID CHAR(5);
cont> DECLARE :SAL INTEGER(2);
cont>
cont> FOR :EMP AS
cont>     SELECT LAST_NAME, EMPLOYEE_ID
cont>     FROM EMPLOYEES
cont>     WHERE EMPLOYEE_ID = '00164'
cont> DO
cont>     SET :ID = :EMP.EMPLOYEE_ID;
cont>     SET :SAL = (SELECT SALARY_AMOUNT
cont>           FROM SALARY_HISTORY
cont>           WHERE EMPLOYEE_ID = :ID
cont>               AND SALARY_END IS NULL);
cont>     TRACE 'EMPLOYEE: ', :ID, ', SALARY: ', :SAL;
cont> END FOR;
cont> END;
~Xt: Employee: 00164, Salary: 0.00
```

The salary should not be zero. This incorrect value is returned because the subquery required the local variable ID which was assigned a value within the FOR loop prior to the subquery. However, this assignment of the ID variable was performed after the subquery had been evaluated.

A workaround to this problem is to reference the FOR loop columns directly using the cursor's handle, rather than taking copies before the subquery is executed.

The correct result is returned when using the FOR loop handle and a direct column reference.

```
SQL> BEGIN
cont> DECLARE :ID CHAR(5);
cont> DECLARE :SAL INTEGER(2);
cont>
cont> FOR :EMP AS
cont>     SELECT LAST_NAME, EMPLOYEE_ID
cont>     FROM EMPLOYEES
cont>     WHERE EMPLOYEE_ID = '00164'
cont> DO
cont>     SET :ID = :EMP.EMPLOYEE_ID;
cont>     SET :SAL = (SELECT SALARY_AMOUNT
cont>           FROM SALARY_HISTORY
cont>           WHERE EMPLOYEE_ID = :EMP.EMPLOYEE_ID
cont>               AND SALARY_END IS NULL);
cont>     TRACE 'Employee: ', :id, ', Salary: ', :sal;
cont> END FOR;
cont> END;
~Xt: Employee: 00164, Salary: 51712.00
```

This problem was corrected in Oracle Rdb7 Release 7.0.1.1. Queries, which are dependent on variables and parameters that can change value, are now processed correctly.

It should be noted that displayed strategies from these queries will change with this release. Query outlines generated with prior releases may no longer match the queries in the compound statement, stored procedure, or stored function. For instance, the example above now involves two separate queries instead of the single query generated by the optimizer in prior versions.

2.3 Oracle RMU Errors Fixed

2.3.1 RMU/BACKUP/BLOCK_SIZE Failed with ACCVIO

Bug 521583.

Performing a RMU/BACKUP operation using the /BLOCK_SIZE qualifier resulted in an access violation if the block size value was less than or equal to 4096.

The following example demonstrates the error:

```
$ RMU/BACKUP/BLOCK_SIZE=2048 MF_PERSONNEL MF_PERSONNEL
%SYSTEM-F-ACCVIO, access violation, reason mask=04, virtual address=00000000, PC
=0013FF70, PSL=03C00004
%RMU-F-FATALERR, fatal error on BACKUP
%RMU-F-FTL_BCK, Fatal error for BACKUP operation at 18-AUG-1997 10:40:13.56
```

A workaround for this problem is to specify a value greater than 4096 for the /BLOCK_SIZE. The actual value may vary depending on whether you are backing up to disk or tape.

This problem was corrected in Oracle Rdb7 Release 7.0.1.1.

2.3.2 RMU/BACKUP/AFTER_JOURNAL/EDIT_FILENAME Incorrectly Applied Edit String

Performing an RMU/BACKUP/AFTER_JOURNAL operation using the /EDIT_ FILENAME qualifier resulted in the edit string being incorrectly applied to the backup file's directory component rather than to its filename component.

The following example demonstrates the error:

```
$ RMU/BACKUP/AFTER_JOURNAL/EDIT_FILENAME=("XXX",YEAR,"XXX") -
_$ [.TMP]MF_PERSONNEL [.TMP]MF_PERSONNEL.AIJ
%RMU-F-FILACCERR, error creating AIJ backup file RDB_USER11:[XXX1997XXX.TMP]MF_PERSONNEL.AIJ;
-RMS-E-DNF, directory not found
-SYSTEM-W-NOSUCHFILE, no such file
%RMU-F-FTL_BCK, Fatal error for BACKUP operation at 29-AUG-1997 19:47:27.35
```

This problem was corrected in Oracle Rdb7 Release 7.0.1.1.

2.3.3 RMU/COLLECT OPTIMIZER_STATISTICS Command Failed with a Bugcheck Error

RMU/COLLECT OPTIMIZER_STATISTICS would fail in all prior versions of Oracle Rdb7 if the database contained temporary tables created using the CREATE GLOBAL TEMPORARY TABLE or CREATE LOCAL TEMPORARY TABLE statements. The result was a bugcheck error within the routine PIOFETCH$WITHIN_DB.

This problem was corrected in Oracle Rdb7 Release 7.0.1.1. RMU/COLLECT now ignores temporary tables and views. The only workaround to this problem is to drop the temporary tables before RMU/COLLECT is executed, and replace them afterwards.

2.3.4 RMU/CONVERT/NOCOMMIT Command Could Corrupt Replication Transfers

Bug 507408.

RMU/CONVERT/NOCOMMIT failed when converting a database that was used as a source for replication transfers used by the Replication Option (Data Distributor):

```
%RMU-I-RMUTXT_000, Executing RMU for Oracle Rdb V7.0-01
Are you satisfied with your backup of DISK1:[USER]SOURCE_DB.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 DISK1:[USER]SOURCE_DB.RDB;1 successfully converted
from version V6.1 to V7.0
%RDB-E-NO_META_UPDATE, metadata update failed
-RDMS-F-NOMETAUPD, metadata updates are prohibited until CONVERT is COMMITTED
```

The error happened as CONVERT tried adding a new index on the system table RDB$TRANSFER_RELATIONS, used by the Replication Option. Because of the failure, the index was not created, but it was critical for the correct operation of the Replication Option. In its absence, the database stopped logging changes to the tables subject to replication, and the target database(s) was therefore no longer kept up to date.

In addition, the initial execution of any newly defined replication transfer failed with bugcheck errors producing exception messages such as:

```
***** Exception at 0057B70B: RDMS$$DML$READY+000000F4
%SYSTEM-F-ACCVIO, access violation, reason mask=00, virtual
address=00000094, PC=0057B70B, PSL=01400004
```

There is no known workaround once updates have been performed against any of the tables subject to replication. If no updates have been performed yet, then you should:

- Rollback the conversion using the RMU/CONVERT/ROLLBACK command.
- Perform the conversion again using the RMU/CONVERT/COMMIT command.

If updates have been performed, then you need to drop the transfer and redefine it. The first execution of the replication transfer will then perform a full re-initialization of the target tables.

Note that there was no problem with the extraction transfers.

This problem was corrected in Oracle Rdb7 Release 7.0.1.1.

2.3.5 RMU/SHOW STATISTIC Command with /INPUT Qualifier Caused a Bugcheck Error

Using the RMU/SHOW STATISTIC command with the /INPUT qualifier caused a bugcheck error dump file to be generated.

The following example shows this problem:

```
$ RMU/SHOW STATISTIC/INP=STATS.DAT
%SYSTEM-F-ACCVIO, access violation, reason mask=00, virtual
address=000001E0, P=00240D05, PSL=03C00000
%RMU-F-FATALOSI, Fatal error from the Operating System Interface.
%RMU-I-BUGCHKDMP, generating bugcheck dump file USER1:[KLEIN]RMUBUGCHK.DMP;
%RMU-F-FTL_SHOW, Fatal error for SHOW operation at 11-JUL-1997 09:45:39.70
```

There is no workaround for this problem.

This problem was corrected in Oracle Rdb7 Release 7.0.1.1.

2.3.6 ENQCNT Greater than 9,999,999 Displayed Incorrectly by RMU/SHOW STATISTICS

The Oracle Rdb7 RMU/SHOW STATISTICS utility was unable to correctly display process ENQCNT values greater than 9,999,999. This problem surfaced because OpenVMS version 7.1 increased the maximum value of the ENQLM process quota to 16,776,959 locks.

This problem was corrected in Oracle Rdb7 Release 7.0.1.1. The RMU/SHOW STATISTICS utility is now able to display ENQCNT values up to 99,999,999.

2.3.7 RMU/SHOW STATISTIC User-Defined Events Not Working with the /NOINTERACTIVE Qualifier

The RMU/SHOW STATISTIC utility User-Defined Events did not work when the /NOINTERACTIVE qualifier was specified.

There is no workaround to this problem other than not using the /NOINTERACTIVE qualifier.

This problem was corrected in Oracle Rdb7 Release 7.0.1.1.

2.3.8 RMU/SHOW STATISTIC "File Locking Statistics" Screen Missing Statistics

The RMU/SHOW STATISTIC utility File Locking Statistics screen did not display values for the “rqsts stalled”, “rqst timeouts”, and other statistics.

The following example shows the affected statistics values not being properly reported:

```
Node: NYMEXE (1/1/1)    Oracle Rdb V7.0-1 Perf. Monitor  28-AUG-1997 09:11:31.31
Rate: 3.00 Seconds          File Locking Statistics         Elapsed: 00:42:48.36
Page: 1 of 1 TMS_DATABASE_ROOT:[DATABASE]CLEARING21_DATABASE.RDB;4  Mode: Online

                        For File: All data/snap files
statistic........... rate.per.second.............. total....... average......
name................ max..... cur..... avg....... count....... per.trans....
locks requested          306      149       44.5       114425           0.2
 rqsts not queued          0        0        0.0            0           0.0
 rqsts stalled             0        0        0.0            0           0.0
 rqst timeouts             0        0        0.0            0           0.0
 rqst deadlocks            0        0        0.0            0           0.0
locks promoted          1184      501       93.2       239389           0.6
 proms not queued          0        0        0.0            0           0.0
 proms stalled             0        0        0.0            0           0.0
 prom timeouts             0        0        0.0            0           0.0
 prom deadlocks            0        0        0.0            0           0.0
locks demoted           1061      613      104.9       269600           0.7
locks released           272      140       31.2        80176           0.2
blocking ASTs             52       24        5.4        13948           0.0
stall time x100          496      260       58.8       151188           0.3


------------------------------------------------------------------------------
Exit Graph Help Menu Options Reset Set_rate Write !
```

There is no workaround for this problem.

This problem was corrected in Oracle Rdb7 Release 7.0.1.1.

2.3.9 RMU/SHOW STATISTIC "Transaction Duration" Collection Duration Too Short

The RMU/SHOW STATISTIC utility Transaction Duration screen collection duration, from 0 to 15 seconds, was often too short for most applications, especially online order-entry and batch transactions which could last for several minutes.

Consider the following Transaction Duration screen. The 15+++ category did not provide much insight into how long those transactions really were.

```
Node: ALPHA3 (1/1/2)    Oracle Rdb X7.0-00 Perf. Monitor  8-SEP-1997 12:56:11.11
Rate: 0.50 Seconds        Transaction Duration (Total)      Elapsed: 03:53:22.03
Page: 1 of 1    KODH$:[R_ANDERSON.WORK.STATS]MF_PERSONNEL.RDB;1     Mode: Online
--------------------------------------------------------------------------------
Transaction rate (per second):   current =     0      average = 2.3
Transaction duration (seconds):  average = 1.37       95th pctile = 3.00
Transaction count:               total =      32879   15+++ =          36
             Scaled distribution of transaction lengths (in seconds)
+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+---+
| *  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |
| *  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |
| ** |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |
| ***|    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |
| **** |   |    |    |    |    |    |    |    |    |    |    |    |    |    |   |
| **** |   |    |    |    |    |    |    |    |    |    |    |    |    |    |   |
| **** |   |    |    |    |    |    |    |    |    |    |    |    |    |    |   |
| ****** |  |    |    |    |    |    |    |    |    |    |    |    |    |    |   |
* ******* |  |    |    |    |    |    |    |    |    |    |    |    |    |    |   |
************  *    *    *    *    *    *    *    *    *    *    *    |    *   |
+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+---+
0....1....2....3....4....5....6....7....8....9....10...11...12...13...14...15+++
                    (Each "*" represents 774 transactions)

--------------------------------------------------------------------------------
Config Exit Help Menu Numbers Options Reset Set_rate Write !
```

There is no workaround to this problem.

This problem was corrected in Oracle Rdb7 Release 7.0.1.1. The RMU/SHOW STATISTIC utility Transaction Duration screen has been enhanced to display transaction durations in either 0 to 15 seconds, or 0 to 15 minutes.

The 0 to 15 second display is the existing Transaction Duration screen; this is the default display.

Using the Config onscreen-menu option, select the Long-duration Transaction Display option to display minutes and the Short-duration Transaction Display option to display seconds.

The Transaction Duration screen can be displayed for read-only, read/write, or both types of transactions.

The Transaction Duration screen can also be selected using the configuration variable TX_DISPLAY and specifying one of the following keywords for its value: SECONDS or MINUTES.

The Transaction Duration screen in both display formats is available during replay of binary output files.

The Transaction Duration screen in both display formats is available cluster-wide.

Consider a database collection period that contains 36 transactions that exceeded the maximum 15 second collection interval. The graphical display is:

```
Node: ALPHA3 (1/1/2)    Oracle Rdb X7.0-00 Perf. Monitor  8-SEP-1997 12:56:11.11
Rate: 0.50 Seconds        Transaction Duration (Total)      Elapsed: 03:53:22.03
Page: 1 of 1    KODH$:[R_ANDERSON.WORK.STATS]MF_PERSONNEL.RDB;1     Mode: Online
--------------------------------------------------------------------------------
Transaction rate (per second):   current =     0      average = 2.3
Transaction duration (seconds):  average = 1.37       95th pctile = 3.00
Transaction count:               total =      32879   15+++ =         36
             Scaled distribution of transaction lengths (in seconds)
+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+---+
| *  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |
| *  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |
| ** |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |
| ***|    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |
| **** |  |    |    |    |    |    |    |    |    |    |    |    |    |    |   |
| **** |  |    |    |    |    |    |    |    |    |    |    |    |    |    |   |
| **** |  |    |    |    |    |    |    |    |    |    |    |    |    |    |   |
| ****** |  |    |    |    |    |    |    |    |    |    |    |    |    |    |   |
* ******* |  |    |    |    |    |    |    |    |    |    |    |    |    |    |   |
************  *    *    *    *    *    *    *    *    *    *    *    *    *   |
+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+---+
0....1....2....3....4....5....6....7....8....9....10...11...12...13...14...15+++
                     (Each "*" represents 774 transactions)

--------------------------------------------------------------------------------
Config Exit Help Menu Numbers Options Reset Set_rate Write !
```

The numeric display of the same information is:

```
Node: ALPHA3 (1/1/2)    Oracle Rdb X7.0-00 Perf. Monitor  8-SEP-1997 13:04:57.93
Rate: 0.50 Seconds        Transaction Duration (Total)      Elapsed: 04:02:08.85
Page: 1 of 1    KODH$:[R_ANDERSON.WORK.STATS]MF_PERSONNEL.RDB;1     Mode: Online
--------------------------------------------------------------------------------
Total transaction count:      32880
Seconds    Tx.Count:   %  #Complete:   % #Incomplete:   %
 0-< 1:       20427  62%      20427  62%       12453  38%
 1-< 2:        9748  29%      30175  91%        2705   9% <- average = 1.37
 2-< 3:        1913   5%      32088  97%         792   3%
 3-< 4:         504   1%      32592  99%         288   1% <- 95th %ile = 3.00
 4-< 5:         119   0%      32711  99%         169   1%
 5-< 6:          49   0%      32760  99%         120   1%
 6-< 7:          28   0%      32788  99%          92   1%
 7-< 8:          14   0%      32802  99%          78   1%
 8-< 9:          17   0%      32819  99%          61   1%
 9-<10:           7   0%      32826  99%          54   1%
10-<11:          10   0%      32836  99%          44   1%
11-<12:           3   0%      32839  99%          41   1%
12-<13:           3   0%      32842  99%          38   1%
13-<14:           2   0%      32844  99%          36   1%
14-<15:           0   0%      32844  99%          36   1%
 15+++:          36   0%      32880 100%           0   0%

--------------------------------------------------------------------------------
Config Exit Graph Help Menu Options Reset Set_rate Write !
```

Selecting the Long-duration Transaction Display option displays the same information in graphical format:

```
Node: ALPHA3 (1/1/2)    Oracle Rdb X7.0-00 Perf. Monitor  8-SEP-1997 13:06:32.03
Rate: 0.50 Seconds       Transaction Duration (Total)      Elapsed: 04:03:42.95
Page: 1 of 1    KODH$:[R_ANDERSON.WORK.STATS]MF_PERSONNEL.RDB;1     Mode: Online
--------------------------------------------------------------------------------
Transaction rate (per minute):   current =     0      average = 2.2
Transaction duration (minutes):  average = 0.02       95th pctile = 1.00
Transaction count:               total =      32880   15+++ =          5
             Scaled distribution of transaction lengths (in minutes)
+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+---+
*****|    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |
*****|    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |
*****|    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |
*****|    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |
*****|    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |
*****|    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |
*****|    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |
*****|    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |
*****|    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |
******    *    |    *    |    |    *    *    |    |    |    |    |    |    *   |
+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+---+
0....1....2....3....4....5....6....7....8....9....10...11...12...13...14...15+++
                      (Each "*" represents 658 transactions)

--------------------------------------------------------------------------------
Config Exit Help Menu Numbers Options Reset Set_rate Write !
```

This information is available in numeric format as well:

```
Node: ALPHA3 (1/1/2)    Oracle Rdb X7.0-00 Perf. Monitor  8-SEP-1997 13:07:22.05
Rate: 0.50 Seconds       Transaction Duration (Total)      Elapsed: 04:04:32.97
Page: 1 of 1    KODH$:[R_ANDERSON.WORK.STATS]MF_PERSONNEL.RDB;1     Mode: Online
--------------------------------------------------------------------------------
Total transaction count:      32880
Minutes    Tx.Count:   %  #Complete:   % #Incomplete:   %
 0-< 1:       32852  99%      32852  99%          28   1% <- average = 0.02
 1-< 2:           5   0%      32857  99%          23   1% <- 95th %ile = 1.00
 2-< 3:          12   0%      32869  99%          11   1%
 3-< 4:           0   0%      32869  99%          11   1%
 4-< 5:           1   0%      32870  99%          10   1%
 5-< 6:           0   0%      32870  99%          10   1%
 6-< 7:           0   0%      32870  99%          10   1%
 7-< 8:           4   0%      32874  99%           6   1%
 8-< 9:           1   0%      32875  99%           5   1%
 9-<10:           0   0%      32875  99%           5   1%
10-<11:           0   0%      32875  99%           5   1%
11-<12:           0   0%      32875  99%           5   1%
12-<13:           0   0%      32875  99%           5   1%
13-<14:           0   0%      32875  99%           5   1%
14-<15:           0   0%      32875  99%           5   1%
 15+++:           5   0%      32880 100%           0   0%

--------------------------------------------------------------------------------
Config Exit Graph Help Menu Options Reset Set_rate Write !
```

As the new display shows, most of the transactions are still under one minute, and only 15 transactions exceeded the maximum 15 minute collection interval.

2.3.10 RMU/SHOW STATISTIC Utility Attached to Database as Application User

The introduction of logical area statistics to the RMU/SHOW STATISTIC utility in Rdb7 Release 7.0.1 required that the utility attach to the database normally. This means it occupied one process slot and reserved the same number of buffers as applications processes would.

In prior versions, the RMU/SHOW STATISTIC utility attached to the database as a “special” user and only reserved two buffers.

This was not a serious problem when using local buffers, but could be quite restricting when using global buffers, especially since the RMU/SHOW STATISTIC utility did not actually use the buffers once it finished attaching to the database.

The workaround to this problem is to define the RDM$BIND_BUFFERS logical to the value "2" before starting the RMU/SHOW STATISTIC utility, or allocate more global buffers for each database.

This problem was corrected in Oracle Rdb7 Release 7.0.1.1. The RMU/SHOW STATISTIC restriction of having to attach to the database normally has been removed; the utility now reserves only 2 buffers and does not require a process slot.

One consequence of removing this restriction is that the RMU/SHOW STATISTIC utility no longer is displayed in the per-process screens.

3

Software Errors Fixed in Oracle Rdb7 Release 7.0.1.2

This chapter describes software errors that are fixed by Oracle Rdb7 Release 7.0.1.2.

3.1 Software Errors Fixed That Apply to All Interfaces

3.1.1 VARCHAR and Numeric Comparison Could Return Incorrect Number of Rows

Bugs 583916 and 606479.

A problem appeared in Oracle Rdb7 Release 7.0.1.1 which caused comparisons between numeric and VARCHAR text variables to fail. The following example shows a query which references a numeric column comparing it to a VARCHAR variable. It should return one row.

```
SQL> CREATE TABLE TT (A INTEGER);
SQL> INSERT INTO TT (a) VALUES (11);
1 row inserted
SQL> DECLARE :X LONG VARCHAR;
SQL> BEGIN
cont> SET :X = '11';
cont> END;
SQL> SELECT A FROM TT WHERE A = :X;
0 rows selected
SQL> rollback;
```

The problem occurred when the numeric string was converted to an intermediate text string which could be compared to the VARCHAR variable, parameter or column. Unfortunately, the intermediate text string was not sized correctly and therefore part of the text version of the numeric value was truncated.

The workaround is to use the CAST function to explicitly convert the VARCHAR value to a numeric value.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.2.

3.1.2 Zig-Zag Match with Different Index Key Datatypes Returned Wrong Results

Bug 584642.

The following query returned wrong results:

```
SQL> CREATE TABLE T1 (
cont>    ID   SMALLINT,
cont>    ID2  INTEGER);
```

```
SQL> CREATE TABLE T2 (
cont>    ID   SMALLINT,
cont>    ID2  INTEGER,
cont>    DATM DATE);

SQL> CREATE UNIQUE INDEX T1_NDX ON T1 (ID, ID2);
SQL> CREATE UNIQUE INDEX T2_NDX ON T2 (ID2, ID);

SQL> INSERT INTO T1 VALUES (1100, 142);
SQL> INSERT INTO T1 VALUES (1110, 142);
SQL> INSERT INTO T1 VALUES (1120, 142);

SQL> INSERT INTO T2 VALUES (140, 1110, DATE '1997-01-15');
SQL> INSERT INTO T2 VALUES (141, 1111, DATE '1997-02-15');
SQL> INSERT INTO T2 VALUES (142, 1120, DATE '1997-03-15');

-- This query should return 2 rows

SQL> SELECT T1.*, T2.*
cont>    FROM T1 T1, T2 T2
cont>    WHERE T1.ID  = T2.ID2 AND
cont>   T2.DATM < DATE '1997-10-24' ;
Conjunct
Match
  Outer loop      (zig-zag)
    Conjunct        Get     Retrieval by index of relation T2
      Index name  T2_NDX [0:0]
  Inner loop      (zig-zag)
    Index only retrieval of relation T1
      Index name  T1_NDX [0:0]
  T1.ID        T1.ID2    T2.ID        T2.ID2   T2.DATM
   1120           142      142          1120   1997-03-15
1 rows selected
```

Oracle Rdb7 introduced a new feature where zig-zag match skipping occurred on the outer or onner leg, but the feature didn't properly handle matching index keys of different datatypes, such as in the example above.

During the process of skipping the outer or inner leg, the optimizer checked to see if the previous index scan advanced the index too far down, and needed to advance the other leg accordingly. During the check, it compared the index key of the current leg with the index key of the other, but it failed to recognize the differences in the datatypes.

As a workaround to this problem, disable the zig-zag skipping feature by defining the logical RDMS$DISABLE_ZIGZAG_MATCH to 1.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.2.

3.1.3 Arithmetic Exception (ARITH_EXCEPT) Calculating Cost of Strategy

Bug 588355.

While determining the best strategy for executing a complex select statement, the calculation of the cost could overflow, causing an arithmetic exception. The following example demonstrates the set of errors displayed when this problem occurs.

```
%RDB-E-ARITH_EXCEPT, truncation of a numeric value at runtime
-SYSTEM-F-HPARITH, high performance arithmetic trap, Imask=00000000,
Fmask=00001000, summary=08, PC=0000000000021E2c, PS=0000000B
-SYSTEM-F-FLTOVF, arithmetic trap, floating overflow at PC=0000000000F21E2C,
PS=0000000B
```

This problem has been corrected in Oracle Rdb7 Release 7.0.1.2.

3.1.4 Queries Including GROUP BY Could Return Wrong Results

Bug 598779.

In rare cases in Oracle Rdb7 Release 7.0.1.1, if a GROUP BY clause was specified in conjunction with an aggregate function(COUNT, SUM, AVG, MAX, MIN) over a view that contained a conditional expression (NULLIF, COALESCE, NVL, CASE, DECODE) then the resulting table could show column values displaced by one row.

This change in behavior was a result of a correction included in Oracle Rdb7 Release 7.0.1.1 which unfortunately caused wrong results to be returned.

The following example shows the problem. View1 contains a CASE expression and a SELECT over View1 which contains a SUM function returns the wrong results:

```
CREATE VIEW VIEW1
 (
    JOB_CODE,
    DEPARTMENT_CODE,
    DEPARTMENT_NAME,
    SALARY_AMOUNT
 )
 AS
 SELECT
      JH.JOB_CODE,
      JH.DEPARTMENT_CODE,
       CASE
          WHEN
               (JH.DEPARTMENT_CODE = ' ') THEN 'Unknown'
          ELSE
               (SELECT DEPARTMENT_NAME FROM DEPARTMENTS D
               WHERE D.DEPARTMENT_CODE = JH.DEPARTMENT_CODE)
          END,
      SH.SALARY_AMOUNT
 FROM
         JOB_HISTORY JH, SALARY_HISTORY SH
 WHERE
 JH.EMPLOYEE_ID = SH.EMPLOYEE_ID;

--
-- Here the DEPARTMENT_NAME returned should be Corporate Administration
-- when DEPARTMENT_CODE is ADMN. DEPARTMENT_NAME Electronics Engineering
-- actually belongs on the subsequent row where DEPARTMENT_CODE is ELEL.
--
SELECT DEPARTMENT_CODE, DEPARTMENT_NAME, SUM(SALARY_AMOUNT)
   FROM VIEW1
    GROUP BY DEPARTMENT_CODE,DEPARTMENT_NAME
    LIMIT TO 2 ROWS;
 DEPARTMENT_CODE   DEPARTMENT_NAME
 ADMN              Electronics Engineering                  7460914.00
 ELEL              Large Systems Engineering                    2479659.00
2 rows selected
```

This problem has been corrected in Oracle Rdb7 Release 7.0.1.2.

3.1.5 Incorrect Page Checksum Value of 1

Bug 590366.

Oracle Rdb applications would sporadically encounter checksum errors where an invalid checksum value of 00000001 was stored on the page. The following example shows the error:

```
%RDB-F-IO_ERROR, input or output error
-RDMS-F-CANTREADDBS, error reading pages 1:560-560
-RDMS-F-CHECKSUM, checksum error - computed 43D5B015, page contained 00000001
```

The problem was introduced in a previous version of Oracle Rdb and existed on all platforms. Oracle Rdb would skip the checksum calculation of a modified page if the current checksum value of the page was 1. The reason this check was put in the code was because the RMU/SET CORRUPT command would explicitly set a corrupt page's checksum to 1. However, because Oracle Rdb always recalculates the checksums of modified pages before they are flushed back to disk, the checksum value of 1 would be overwritten with the proper checksum of the page. To avoid this, the check was added.

The checksums of pages were calculated by adding all the longwords on the page. Because the calculation would very likely overflow a longword, the result was as likely to be 1 as any other value. The check mentioned above failed to realize that 1 was a valid checksum. So, if a page's checksum legitimately resulted in a value of 1, the next time that page was modified, its checksum would not be recalculated and the checksum of 1 would remain on the page. Any subsequent access to that page would result in the above error.

There are no known workarounds for this problem. However, because the rest of the page is valid, RMU/ALTER could be used to reset the checksum of the page to its proper value. The following is an example of using RMU/ALTER to fix a bad checksum:

```
$ RMU/ALTER TEST_DB
%RMU-I-ATTACH, now altering database "DISK:[DIR]DB.RDB;1"

RdbALTER> RADIX HEX
RdbALTER> AREA 1 PAGE 560
RdbALTER> VERIFY
%RMU-W-PAGCKSBAD, area RDB$SYSTEM, page 560
                  contains an invalid checksum
                  expected: 43D5B015, found: 00000001
RdbALTER> DEPOSIT CHECKSUM = 43D5B015
RdbALTER> VERIFY
RdbALTER> COMMIT
RdbALTER> EXIT
```

This problem has been corrected in Oracle Rdb7 Release 7.0.1.2.

3.1.6 Query Using Sorted Ranked Index Could Return Wrong Results

Bug 563213.

In Oracle Rdb7, a new type of sorted B-tree index was added, known as a SORTED RANKED index. When the dynamic query optimizer used a SORTED RANKED B-tree, it was possible in prior releases for the "ZeroShortCut" strategy to be taken incorrectly if the isolation level was set to either READ COMMITTED or REPEATABLE READ. This problem did not occur for ISOLATION LEVEL SERIALIZABLE (which is the default).

The "ZeroShortCut" strategy is used when the optimizer expects the query to return zero rows for the query. This is based on information returned during the estimation step (Estim) as shown in the following example.

```
~E#0014.01(1) Estim   Ndx:Lev/Seps/DBKeys 1:0/0\0 2:0/0/0 ZeroShortcut
```

This output from the EXECUTION flags (RDMS$DEBUG_FLAGS "E") shows that zero rows would be returned from this query.

The I/O performed by the estimation step is limited so that it adds as little I/O to the query as possible (usually it results in the loading of the index root node and some of the leaf nodes which are normally needed later during processing). In the problem case, the imposed limit terminated the processing before the estimates were fully calculated and the resulting zero value was assumed to indicate a precise value for the number of matching values. Hence the "ZeroShortCut" optimization was taken.

The algorithm has now been improved to ensure that exceeding these processing limits causes normal correct processing. The estimation step performs index data sampling using I/O up to four times the index depth.

At the same time a new logical name, RDMS$REFINE_RANKED_ESTIMATE, has been created to allow further tuning of the estimation step for sorted ranked indexes. This logical name can be defined as a positive integer which specifies the extra I/O to perform to refine the retrieval estimates based on the cardinalities stored in the leaf nodes of this type of index. The value represents the multiplier for the index depth. For sites with very large (deep) indexes, defining this logical name to 2 or 3 may allow a more precise estimate to be determined. The higher the value, the more I/O will be expended to refine the estimate. Very high values will not be as productive because more I/O might be expended to refine the estimate than would actually be required to fetch the data.

Workarounds include:

- Disabling the estimation step by defining the RDMS$MAX_STABILITY logical name
- Making the index sorted (not a sorted ranked index)

This problem is corrected in Oracle Rdb7 Release 7.0.1.2.

3.1.7 Queries Using Ranked B-Tree Indexes Could Result in Errors

In rare cases, when a ranked B-tree index was used, a bugcheck error could occur showing "PSII2SCANSTARTBBCSCAN" in the error message. This problem could occur when a query performed a reverse scan followed by a forward scan of the same B-tree index within the same transaction. The following example shows this type of transaction.

```
SQL> SELECT * FROM EMPLOYEES WHERE EMPLOYEE_ID < "01000" ORDER BY
cont> EMPLOYEE_ID DESCEND;
   .
   .
   .
cont> SELECT * FROM EMPLOYEES WHERE EMPLOYEE_ID < "01000";
```

This problem does not cause database corruption.

A possible workaround is to use non-ranked B-tree indexes instead.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.2.

3.1.8 Process Deleted when a Hashed Index Was Dropped

Deleting a hashed index that had an extremely large number of duplicates could cause a process to be deleted. This occurred because the deleting operation could exhaust the stack space.

A workaround is to define the logical name, RDMS$BIND_EXEC_STACK_SIZE, to increase the size of the stack.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.2.

3.1.9 Character Conversion Problem Could Cause an Infinite Loop

Bug 562592.

OpenVMS VAX platforms.

If a table was defined with a column of the BIGINT datatype and an index was defined using that column, a looping condition could result if that column was accessed through a SQL expression using character format in a WHERE clause.

The following example defines the table and index:

```
SQL> CREATE TABLE T (COL1 INTEGER, COL2 BIGINT  . . . );
SQL> CREATE INDEX I ON T (COL1, COL2  . . . );
```

The following query accesses the column using character format:

```
SQL> SELECT * FROM T WHERE ( . . .  COL2 = '0' . . . );
```

Accessing a row through the index should cause the character '0' to be converted to a QUADWORD but instead may have resulted in an infinite loop.

A workaround is to use the numeric format of 0 instead of the character format in all queries or remove the column from the index.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.2.

3.1.10 TSNBLK Locks Acquired Too Frequently During Transaction Startup and Commit

The TSNBLK lock was acquired too often during the transaction start or commit sequence. This caused excessive stalls on the lock and, eventually, a significant degradation in performance.

The problem was magnified with Oracle Rdb7 because the number of entries in the TSNBLK data structure was decreased from 50 to 28. This resulted in a near doubling of the number of TSNBLK entries in the database rootfile.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.2. The algorithm which manages the TSNBLK has been redesigned and optimized for efficient processing.

For example, using an unmodified MF_PERSONNEL database with one process inserting 8,192 records and one insert per transaction, results in a reduction of 122,895 TSNBLK lock requests. This is an average reduction of 14.9 locks per transaction.

Using the MF_PERSONNEL database configured for 2,048 users on 16 nodes, 67,633 records were inserted into the database using five processes, 1 record insert per transaction. All inserts occurred on a single node. The number of TSNBLK lock requests saved using the new algorithm was 2,723,265. The reduction in TSNBLK lock requests is approximately 66 per transaction!

Of course, this performance increase may not be reflected in your particular database environment.

3.1.11 Hold Cursor Closed on Set Transaction

Bug 558308.

A hold cursor in a precompiled SQL program or a SQL module language program could be closed after the execution of a SET TRANSACTION statement.

The following C and SQLMOD modules demonstrate this problem. The program starts a transaction, opens a hold cursor, fetches and then commits. It then starts a transaction which results in the cursor being closed from the program's point of view (it is still open from Rdb's point of view). The fetch then fails because it believes that the cursor was closed.

```
#INCLUDE "STDIO.H"

EXEC SQL INCLUDE SQLCA;
MAIN () {

CHAR    SNAME[210], A[100], B[100], C[100];
INT     TRANSACTION_FLAG;

BILL_START_RW_TRANSACTION(&SQLCA, &TRANSACTION_FLAG);
PRINTF("Set Transaction: %D\N", SQLCA.SQLCODE);

BILL_OPEN_BILLING_CURSOR(&SQLCA);
PRINTF("Open: %D\N", SQLCA.SQLCODE);

FETCH_BILLING_CURSOR(&SQLCA, A, B, C);
PRINTF("Fetch: %D\N", SQLCA.SQLCODE);

BILL_COMMIT_TRANSACTION(&SQLCA);
PRINTF("Commit: %D\N", SQLCA.SQLCODE);

BILL_START_RW_TRANSACTION(&SQLCA, &TRANSACTION_FLAG);
PRINTF("Set Transaction: %D\N", SQLCA.SQLCODE);

FETCH_BILLING_CURSOR(&SQLCA, a, b, c);
PRINTF("Fetch: %D\N", SQLCA.SQLCODE);

BILL_COMMIT_TRANSACTION(&SQLCA);
PRINTF("Commit: %D\N", SQLCA.SQLCODE);
}

==========

MODULE          FOO                         -- Module name goes here
LANGUAGE        GENERAL                     -- Language of calling program
PARAMETER       COLONS

DECLARE SCP_DB_HANDLE ALIAS FOR FILENAME 'MF_PERSONNEL.RDB'
                  DBKEY SCOPE IS ATTACH

DECLARE BILLING_CURSOR TABLE CURSOR WITH HOLD PRESERVE ON COMMIT FOR
    SELECT EMPLOYEE_ID, LAST_NAME, FIRST_NAME FROM SCP_DB_HANDLE.EMPLOYEES
        E WHERE E.EMPLOYEE_ID > '0'

------------------------------------------------------------------------
PROCEDURE BILL_START_RW_TRANSACTION
SQLCA,
:P_LOCAL_TRANSACTION_STARTED    INTEGER;

BEGIN
DECLARE :SQL_DETECT_TRANS_STATE INTEGER;
GET DIAGNOSTICS :SQL_DETECT_TRANS_STATE = TRANSACTION_ACTIVE;
TRACE 'TRANSACTION STATE WAS ', :SQL_DETECT_TRANS_STATE;
```

```
IF :SQL_DETECT_TRANS_STATE = 0
THEN
    SET TRANSACTION READ WRITE RESERVING SCP_DB_HANDLE.EMPLOYEES FOR
        SHARED WRITE;
    SET :P_LOCAL_TRANSACTION_STARTED = -1;
ELSE
    SET :P_LOCAL_TRANSACTION_STARTED = 0;
END IF;
END;
------------------------------------------------------------------------
PROCEDURE BILL_COMMIT_TRANSACTION
SQLCA;

COMMIT;
------------------------------------------------------------------------
PROCEDURE BILL_ROLLBACK_RW_TRANSACTION
SQLCA;

ROLLBACK;
------------------------------------------------------------------------
PROCEDURE BILL_OPEN_BILLING_CURSOR
SQLCA;

OPEN BILLING_CURSOR;
------------------------------------------------------------------------
PROCEDURE BILL_CLOSE_BILLING_CURSOR
SQLCA;

CLOSE BILLING_CURSOR;
------------------------------------------------------------------------
PROCEDURE FETCH_BILLING_CURSOR
SQLCA,
:P_EMPLOYEE_ID      CHAR(5),
:P_LAST_NAME        CHAR(14),
:P_FIRST_NAME       CHAR(10);

FETCH BILLING_CURSOR INTO :P_EMPLOYEE_ID, :P_LAST_NAME ,:P_FIRST_NAME;
```

A workaround is not to execute a SET TRANSACTION command between transactions. Let the declared transaction be used instead.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.2.

3.1.12 Index Size Restricted Incorrectly for Collating Sequences

Bug 586079.

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

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

These extra bytes were not being taken into account when calculating the maximum index key size during index creation. Rather than checking the index key length, the length of the key being encoded was checked. Therefore, indexes whose length was in excess of 255 bytes were permitted. This led to unexpected errors when the index was later used.

The following example demonstrates a problem evaluating a constraint while inserting data into a table. The constraint is evaluated using an index on a 233 character column, with a German collating sequence, where the index creation should have failed, due to the 255 byte size limit.

```
SQL> CREATE DATABASE
cont>      FILENAME 'TESTDB.RDB'
cont>      COLLATING SEQUENCE GERMAN GERMAN;
SQL> CREATE TABLE EMPLOYEE_INFO (
cont>      EMP_NAME CHAR (233),
cont>      CONSTRAINT EMP_NAME_PK
cont>         PRIMARY KEY (EMP_NAME)  NOT DEFERRABLE);
SQL> CREATE INDEX EMP_NAME_IDX
cont>      ON EMPLOYEE_INFO (
cont>      EMP_NAME     ASC)
cont>      TYPE IS SORTED;
SQL> COMMIT;
SQL> INSERT INTO EMPLOYEE_INFO (EMP_NAME) VALUES
cont> ('12345678901234567890123456789012345678901234567890123456789012345678901234567890');
%RDB-E-INTEG_FAIL, violation of constraint EMP_NAME_PK caused operation to fail
-RDB-F-ON_DB, on database USER4:[WORK]TESTDB.RDB
```

To work around this problem, use the formula described above to determine the actual size of the index and change the definition of the index to be within the limit of 255 bytes.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.2.

3.1.13 Bugcheck Error when Committing a Delete from a Temporary Table

Bug 575037.

In prior releases of Oracle Rdb7, it was possible, under certain circumstances, to generate a bugcheck error when committing deleted rows from a temporary table.

The following example displays a sequence of statements that previously would result in a bugcheck error at commit time:

```
SQL> ATTACH 'FILENAME PERSONNEL';
SQL> INSERT INTO GLOBAL_TEMP_A(NAME, COUNT) VALUES ('Jones',1);
SQL> UPDATE GLOBAL_TEMP_A SET COUNT = 2  WHERE NAME = 'Jones';
SQL> DELETE FROM GTB_A;
SQL> COMMIT;
```

A workaround to this problem is to keep the DELETE and TRUNCATE TABLE statements in separate transactions from UPDATE statements for the same rows.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.2.

3.1.14 Monitor Process Quotas Increased

All OpenVMS platforms.

When an Oracle Rdb7 monitor (RDMMON) process is started using the RMU Monitor Start command, the quota limits that the monitor process uses are determined as the larger of three factors:

- A hard-coded "minimum-necessary" value.
- The quota value from the user designated by the RDM$MON_USERNAME logical name (with a default value of "SYSTEM").
- The quota value from the process performing the startup.

The minimum values of several of these quotas have been increased in Oracle Rdb7 Release 7.0.1.2. The hard-coded minimum values for each monitor quota is shown in Table 3–1.

Table 3–1 Monitor Process Minimum Quotas

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

These quota value minimums have been adjusted to help the monitor open large databases.

3.1.15 %RDB-F-IO_ERROR, -COSI-F-WRITERR, RMS-F-ISI Errors Fixed

Bug 561516.

In prior releases of Oracle Rdb7, the following error could result if Rdb attempted to use a temporary file on disk for storing intermediate results fetched from the inner loop of a join with the zig-zag match strategy.

```
%RDB-F-IO_ERROR, input or output error
-COSI-F-WRITERR, write error
-RMS-F-ISI, invalid internal stream identifier (ISI) value
```

This error occurred when the total row size exceeded the current setting for RDMS$BIND_WORK_VM (the default is 10000 bytes if no logical name is defined). This logical name defines the amount of virtual memory used for holding temporary results. When it is full, the data overflows into a temporary disk file whose location is defined by the RDMS$BIND_WORK_FILE logical name.

During zig-zag match strategy, when Rdb was not able to cache any rows, it immediately attempted to write to the work file before having created the temporary file.

The workaround to this problem is to define the logical name, RDMS$BIND_WORK_VM, to a size which can accommodate one row. Allocating more virtual memory to the work area will improve overall query performance because the temporary results will be saved in memory and incur little or no disk I/O.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.2.

3.1.16 Second Online Snapshot Truncation Could Wait Indefinitely

Bug 561351.

A process doing a second consecutive online snapshot truncation could wait indefinitely with the "waiting for snap truncation L1 (PR)" message displaying when another process was attached to the database.

The workaround is to disconnect the blocking process from the database.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.2.

3.1.17 Operator Notification Frequency of AIJ Fullness

Bug 574104.

All OpenVMS platforms.

When enabled, Oracle Rdb notifies the system operator when the last available AIJ file is more than 90% full. This notification occurred on each write to the AIJ file once the AIJ file reached 90% fullness. On some systems, the volume of operator notification messages could make it difficult to log in and take corrective action.

This situation has been improved in Oracle Rdb7 Release 7.0.1.2. Oracle Rdb now attempts to perform this notification no more than once per minute per node. This should reduce the volume of operator messages while continuing to notify the operator of the situation.

3.1.18 Creating a Ranked B-tree Index On a Large Table Could Fail

Bug 568672.

On rare occasions, creating a ranked B-tree index could fail due to overflowing the node size.

Creating an index such as the following could result in a bugcheck error.

```
SQL>   CREATE INDEX  INDEX1
cont>   ON  TABLE1
cont>      ( COL1
cont>       ,COL2
cont>       ,COL3
cont>       ,COL4
cont>       ,COL5
cont>       ,COL6
cont>       ,COL7
cont>       ,COL8
cont>       ,COL9
cont>       ,COL10)
cont>   TYPE IS SORTED RANKED
cont>         DUPLICATES ARE COMPRESSED
cont>   STORE IN INDEX1;
```

As a possible workaround for this problem, set a large node size, such as 1000, or use a regular B-tree (non-ranked) index.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.2.

3.1.19 RMU/MOVE_AREA Command Failed to Delete Files

Bug 463025.

In certain cases, the RMU/MOVE_AREA/ONLINE command could fail to correctly delete the old storage area files because they were still locked by user processes. Further, the user processes could continue to access the existing storage area files.

The following example demonstrates the failure of the user process (running interactive SQL) to close the storage area files that were moved.

```
SQL> ATTACH 'FILENAME MF_PERSONNEL';
SQL> UPDATE EMPLOYEES SET MIDDLE_INITIAL = 'Q';
100 rows updated
SQL> COMMIT;
SQL> $ RMU/MOVE/ONLINE/DIR=DUA0:[DB.X] MF_PERSONNEL EMPIDS_MID
%RMU-I-QUIETPT, waiting for database quiet point
%RMU-I-RELQUIETPT, Database quiet point lock has been released.
%RMU-I-MOVTXT_01, Moved storage area DUA0:[DB.X]EMPIDS_MID.RDA;1
%RMU-I-RESTXT_05,     rebuilt 1 space management page
%RMU-I-MOVTXT_02,     moved 0 inventory pages
%RMU-I-RESTXT_07,     rebuilt 0 logical area bitmap pages
%RMU-I-MOVTXT_03,     moved 51 data pages
%RMU-I-RESTXT_01, Initialized snapshot file DUA0:[DB.X]EMPIDS_MID.SNP;1
%RMU-I-LOGINIFIL,     contains 109 pages, each page is 2 blocks long
%RMU-F-CANTDELETE, error deleting "DUA0:[DB]EMPIDS_MID.RDA;1"
%COSI-E-FLK, file currently locked by another user
-RMS-E-FLK, file currently locked by another user
%RMU-F-CANTDELETE, error deleting "DUA0:[DB]EMPIDS_MID.SNP;1"
%COSI-E-FLK, file currently locked by another user
-RMS-E-FLK, file currently locked by another user
```

As a workaround, do not use the /ONLINE qualifier.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.2. Oracle Rdb7 now correctly closes the storage area files that have been moved.

3.1.20 RMU Online Backup Locked Snapshot Pages for Long Durations

Bug 547583.

In some large, complex databases with many storage area files, RMU backup online operations could lock snapshot pages for long durations (up to several minutes) before the pages were released.

As a possible workaround, consider using a smaller number of buffers for the backup operation.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.2. RMU now unlocks and releases snapshot pages from the buffer pool as soon as possible after they have been used rather than waiting for the buffer to be flushed from the buffer pool by other snapshot pages. A possible side effect of this change is that some backup operations may incur a small increase in I/O to snapshot storage areas. However, this increase should be small enough that it will not impact the duration of the backup operation and comes with the gain of improved application performance due to shorter locking by the backup.

3.1.21 Default Recovery Unit Journal Location Could be Incorrect when Restoring Database to Another System

Bug 458415.

A database restored to another system could fail to work correctly if the database default recovery unit journal (RUJ) file location did not exist.

In the following example the system A database had a default RUJ file location of DUA0:[RUJ]. When this database was restored on system B, DUA0 was an invalid device. Because of this, it was not directly possible to alter the database to change the default location because this operation required an RUJ file to be created and it can not be created because the current location is invalid.

On system A:

```
$ SQL
SQL> ALTER DATABASE DB RECOVER JOURNAL (LOCATION IS 'DUA0:[RUJ]');
$ RMU/BACKUP DB DBBCK
```

On system B (where disk DUA0: doesn't exist):

```
$ RMU/RESTORE/NOCDD/DIRECTORY= <newdir> DBBCK
$ SQL
SQL> ALTER DATABASE FILENAME DB RECOVER JOURNAL (NO LOCATION);
RDB-F-SYS_REQUEST, error from system services request
-RDMS-F-FILACCERR, error parsing file DUA0:[RUJ]DB$0001038E62B3.RUJ
-COSI-I-NOTDISKFILE, file is not a disk file
```

As a workaround, define a logical name for the missing device (DUA0 in the example) to point to an existing device on the system.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.2. If there is a problem parsing the RUJ file location when creating the RUJ file, the file is parsed again without the database default RUJ file location.

Because the default RUJ file location may be ignored, the Oracle Rdb Monitor (RDMMON) will attempt to validate the default RUJ file location for the database when the database is opened. If the monitor detects a problem with the default RUJ file location, information is written into the monitor log file indicating the problem.

3.1.22 RDB-E-OBSOLETE_METADA, RDMS-E-RTNNEXTS Error when Calling an External Function from a Constraint

Bug 556722.

In some cases, when calling an external function from within a constraint, the following error could be returned:

```
%RDB-E-OBSOLETE_METADA, request references metadata objects that no longer exist
-RDMS-E-RTNNEXTS, routine does not exist in this database.
```

When a table is updated, Rdb loads and analyzes all constraints defined for that table, or constraints which reference that table such as FOREIGN KEY or CHECK constraints. Based upon the operation (INSERT, UPDATE, or DELETE) some of these constraints may be discarded because they are not required for the current operation. In the case of the reported problem, the failure occurred because of an attempt to process an external function referenced only by the discarded constraint.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.2.

3.1.23 Left Outer Join Query with a View Containing Sub-Select and Union Caused a Bugcheck Error

Bug 555114.

The following left outer join query with a view containing a sub-select and union caused a bugcheck error and stopped the running process:

```
SQL> SELECT BUDGET.DEPARTMENT, DEPARTMENT.CODE
cont>   FROM BUDGET LEFT OUTER JOIN DEPARTMENT ON
cont> BUDGET.DEPARTMENT = DEPARTMENT.CODE ;
```

Where the view BUDGET definition is as follows:

```
SQL>  CREATE VIEW BUDGET (DEPARTMENT) AS
cont>    SELECT
cont> (
cont> SELECT IDENTIFIER FROM DEPARTMENT D
cont>   WHERE D.CODE = O.CODE
cont>   UNION
cont>    SELECT IDENTIFIER FROM ORGANIZATION_STRUCTURE S,
cont>                                DEPARTMENT D
cont>      WHERE O.CODE = S.CHILD_CODE AND
cont>         S.PARENT_CODE = D.CODE)
cont>        FROM COST_CENTER C, ORGANIZATION O
cont>    WHERE
cont>              C.CODE      = O.CODE and
cont>              C.SET_CODE  = O.SET_CODE;
```

The cause of this problem was in the query compilation, where the sub-select query was not properly nested to the left outer join query.

As a workaround to this problem, remove the union in the view query.

```
SQL> DROP VIEW BUDGET;
cont>  CREATE VIEW BUDGET (DEPARTMENT) AS
cont>    SELECT
cont> (
cont> -- Remove the following union leg
cont> --
cont> -- SELECT IDENTIFIER FROM DEPARTMENT D
cont> --        WHERE D.CODE = O.CODE
cont> --        UNION
cont>   SELECT IDENTIFIER FROM ORGANIZATION_STRUCTURE S,
cont>                                DEPARTMENT D
cont>          WHERE O.CODE = S.CHILD_CODE AND
cont>                S.PARENT_CODE = D.CODE
cont>        )
cont>        FROM COST_CENTER C, ORGANIZATION O
cont>         WHERE
cont>              C.CODE      = O.CODE AND
cont>              C.SET_CODE  = O.SET_CODE;
```

This problem has been corrected in Oracle Rdb7 Release 7.0.1.2.

3.1.24 Queries with Constant Equality Predicate Caused Performance Problem

Bug 506504.

Queries using a constant equality predicate, such as the following, caused sequential access index retrieval and poor performance.

```
SQL> SELECT  * FROM TABLEA TA, TABLEB TB WHERE
cont>     (TA.PRIMARY_KEY = TB.PRIMARY_KEY
cont>      AND ( ('Y' = 'Y' AND TA.COL1 = 'AAAAAA') OR  ('N' = 'N') )
cont>      AND ( ('Y' = 'Y' AND TB.COL1 = 'CCCCCCCCCC') OR  ('N' = 'N') )
cont>      AND ( ('Y' = 'Y'  AND    TA.COL2 BETWEEN 6 AND 7) OR  ('N' = 'N') ) );
Solutions tried 39
Solutions blocks created 7
Created solutions pruned 4
Cost of the chosen solution   5.0139539E+02
Cardinality of chosen solution   0.0000000E+00
Cross block of 2 entries
  Cross block entry 1
    Conjunct        Get     Retrieval sequentially of relation TABLEB
  Cross block entry 2
    Leaf#01 FFirst TABLEA Card=23565
      BgrNdx1 TABLEA_1_NDX [1:1] Bool Fan=12
      BgrNdx2 TABLEA_2_NDX [1:1] Bool Fan=12
```

The selectivity cost for an expression like ('Y' = 'Y') was computed as zero and resulted in zero cardinality for the chosen solution. This caused the optimizer to choose sequential retrieval and impacted performance.

As a workaround to this problem, remove those predicates that compare constant values in a SELECT expression as the following example shows:

```
SQL> SELECT  * FROM TABLEA TA, TABLEB TB WHERE
cont>     (TA.PRIMARY_KEY = TB.PRIMARY_KEY
cont>      AND (TA.COL1 = 'AAAAAA' )
cont>      AND (TB.COL1 = 'CCCCCCCCCC' )
cont>      AND (TA.COL2 BETWEEN 6 AND 7 ) );
```

This problem has been corrected in Oracle Rdb7 Release 7.0.1.2.

3.1.25 Queries with Aggregate Subquery and Transitive Predicate Returned Wrong Results

Bug 532257.

The following query, using an aggregate subquery and transitive predicate, returned the wrong number of rows in Oracle Rdb7.

```
SQL> SELECT DISTINCT A.ID, A.DATA, B.OTHER_DATA
cont> FROM TABA A, TABB B
cont>     WHERE (B.ID = A.ID AND
cont>      B.DATA = A.DATA) AND
cont>  EXISTS
cont>      (SELECT DISTINCT C1.DATA1 FROM TABC C1
cont>   WHERE (C1.ID = A.ID AND
cont>          C1.DATA = A.DATA AND
cont>          C1.ID1 =
cont>            (SELECT MAX(C2.ID1) FROM TABC C2
cont>        WHERE C2.ID = C1.ID AND
cont>       C2.DATA = C1.DATA)));
```

Oracle Rdb7 disallowed transitivity, by default, for queries containing a COUNT aggregate subquery. But for queries with aggregate subqueries other than COUNT, such as MAX in the example query, transitivity was allowed by default, and caused the query to return the wrong results.

A workaround to this problem is to disable the transitivity selection feature for queries which contain aggregate subqueries other than COUNT, such as, MIN, MAX, SUM, and AVG. Disable the transitivity selection feature by defining the logical name RDMS$DISABLE_TRANSITIVITY as "YES".

This problem has been corrected in Oracle Rdb7 Release 7.0.1.2.

3.1.26 Optimizer Strategy Returned Wrong Result

Bug 590477.

For the following query the optimizer used a match strategy that returned the wrong result.

```
SQL>  SELECT T3.COL1, TABLE2.COL1, T3.COL2
cont>    FROM ( SELECT DISTINCT COL1
cont>              FROM TABLE1
cont>         ) TABLE2,TABLE3 T3
cont>    WHERE NOT EXISTS ( SELECT *
cont>                          FROM TABLE3 TAB3
cont>                          WHERE TAB3.COL1 = TABLE2.COL1)
cont>      AND T3.COL1 = 'ABC';
Conjunct
Match
  Outer loop
    Cross block of 2 entries
      Cross block entry 1
        Conjunct        Get
        Retrieval by index of relation TABLE3
          Index name  TABLE3_IDX [1:1]
      Cross block entry 2
        Merge of 1 entries
          Merge block entry 1
          Reduce (distinct) Index only retrieval of relation TABLE1
            Index name  TABLE1_IDX1 [0:0]
  Inner loop      (zig-zag)
    Aggregate-F1    Index only retrieval of relation TABLE3
      Index name  TABLE3_IDX [0:0]
```

This problem was caused by the match strategy where the match key of the inner leg was matched against that of the inner block of the cross join at the outer match leg, where no sorting was done to match against the inner leg.

This problem also existed in previous versions of Oracle Rdb, but the problem was hidden by the old cost model which causes the optimizer to choose a three-way cross strategy, whereas, in Oracle Rdb7, the new cost model causes the optimizer to choose the match strategy, and thus exposes the hidden problem.

As a workaround, define the logical name RDMS$USE_OLD_COST_MODEL to YES and run the query to produce a query outline. Reset the logical name and apply the query outline when running the query in the future.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.2.

3.1.27 Bugcheck Error at COSI_MEM_FREE_VMLIST when Detaching from a Database and Statistics Were Enabled

Bug 532516.

When statistics were enabled, bugcheck errors at COSI_MEM_FREE_VMLIST could occur. This problem was introduced in Oracle Rdb7 Release 7.0.1.

The following example shows the command used to cause this problem.

```
$ RMU/LOAD/TRANSACTION=EXCLUSIVE DB_CLH FUNCTIONAL_UNITS -
FUNCTIONAL_UNITS.CLHUNL /FIELDS=( -
FU_NO,FU_NM,FU_NM_ABBREV,FU_CATEGORY_NO,FU_TYPE_NO,FU_NO_REPORTS)
```

The RMU/LOAD command actually completed even though a bugcheck error occurred while disconnecting from the database.

A possible workaround for this problem is to disable statistics collection by defining the logical name RDM$BIND_STATS_ENABLED to "0".

This problem has been corrected in Oracle Rdb7 Release 7.0.1.2.

3.1.28 Idle Processes No Longer Perform Global Checkpoints

Prior to Oracle Rdb7, the behavior of idle processes with respect to global checkpoints was the following:

- Attached inactive processes performed global checkpoint operations.
- Global checkpoint occurred as soon an AIJ switch was made.

However, Rdb7 behavior was different in that it appeared that a global checkpoint never occurred (without ABS) for inactive processes following AIJ switchover.

The following example shows this problem:

```
AIJ#            Session 1      Session 2
1               insert
                commit
                                insert
                                commit
2                               ... until aij_switch
                                insert
                                commit
3                               ... until aij_switch

                 Process1 has checkpoint in AIJ1
                                Process2 has checkpoint in AIJ3
```

The workaround is to manually perform a global checkpoint operation using the RMU/CHECKPOINT utility.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.2. Idle processes now perform global checkpoints as they did in previous versions.

3.1.29 Read-Only Transactions Could Update Database Rootfile and AIJ File

When using databases that were upgraded to Oracle Rdb7 Release 7.0.1, it was possible for read-only transactions to modify the database rootfile and AIJ files. This problem occurred when a read-only transaction detected a larger logical area DBID than any previously seen.

This problem did not occur for databases that were newly created using Oracle Rdb7 Release 7.0.1.

The problem could be detected by examining the AIJ file using the RMU/DUMP /AFTER_JOURNAL utility. The journaled changes were displayed as a TYPE=D record with the entry "KROOT: ID=48 (** UNKNOWN **), LENGTH=2". A diagnostic message will also be displayed; for example "Detected TID change from 3598 to 3925".

This problem did not cause corruption of the live database. However, it did effect recovery of the database, either using the database recovery process ("DBR") or rolling forward the AIJ file using the RMU/RECOVER utility.

The following example shows an occurrence of the problem.

```
   .
   .
   .
18/55            TYPE=D, LENGTH=252, TAD= 8-DEC-1997 01:13:03.77, CSM=00
    TID=23, TSN=0:100298784, AIJBL_START_FLG=1, SEQUENCE=1
        KROOT: ID=48 (** UNKNOWN **), LENGTH=2
        KROOT: ID=48 (** UNKNOWN **), LENGTH=2
        KROOT: ID=48 (** UNKNOWN **), LENGTH=2
        KROOT: ID=48 (** UNKNOWN **), LENGTH=2
        KROOT: ID=48 (** UNKNOWN **), LENGTH=2
18/55            TYPE=D, LENGTH=28, TAD= 8-DEC-1997 01:17:21.50, CSM=00
    TID=3925, TSN=0:100298862, AIJBL_START_FLG=1, SEQUENCE=1
    Detected TID change from 3598 to 3925
        KROOT: ID=48 (** UNKNOWN **), LENGTH=2

18/56            TYPE=R, LENGTH=14, TAD= 8-DEC-1997 01:17:21.50, CSM=00
    TID=3925, TSN=0:100298862, AIJBL_START_FLG=1, SEQUENCE=2
    .
    .
    .
```

There is no workaround for this problem. However, it is sometimes possible to prevent the problem by manually creating a B-tree or hash index using a read/write transaction, then deleting the index. Be sure to commit the transaction.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.2. Read-only transactions no longer modify the database rootfile and AIJ journal.

3.2 SQL Errors Fixed

3.2.1 DECLARE TABLE Statement Could Cause a Bugcheck Error

A DECLARE TABLE statement could cause a bugcheck error when defining columns for the table. This problem occurred using SQLPRE and COBOL. The bugcheck occurred at SQL$$CREATE_SQL_FIELD + 000002C9.

This problem was not always reproducible.

This problem has been corrected in Oracle Rdb7 Releas 7.0.1.2.

3.2.2 Unexpected Errors when Calling Stored Procedures

Bugs 593434 and 598038.

In Oracle Rdb7 Release 7.0.1.1 a problem was introduced which could cause a bugcheck error or another unexpected error when running a stored procedure. Two of the reported errors are shown below.

Example 3–1 Bugcheck in RDMS$$PRE_EXECUTION (Alpha OpenVMS)

```
***** Exception at 00FF2C20 : RDMS$$PRE_EXECUTION + 000002E0
%SYSTEM-F-ACCVIO, access violation, reason mask=00, virtual address=000000000
000002C, PC=0000000000FF2C20, PS=0000000B
```

Example 3–2 Unexpected ARITH_EXCEPT exception (VAX OpenVMS)

```
%RDB-E-ARITH_EXCEPT, truncation of a numeric value at runtime
-SYSTEM-F-ROPRAND, reserved operand fault at PC=0068EA3A, PSL=01C00000
```

For this problem to occur, the stored procedure, or stored function must have had in excess of 64 parameters and/or variables in the routine and these variables must have been OUT parameters, INOUT parameters, or updatable variables. For instance, a routine with 65 parameters which updated the last parameter may have encountered this problem. Another example would involve a stored routine with 10 parameters and in excess of 54 variables and one of the last declared variables was updated.

The problem could also occur within a multistatement procedure (an anonymous non-stored routine) which had in excess of 64 variables declared.

If the count of the variables and parameters of the routine was less than or equal to 64 then this problem did not occur.

This problem has been corrected in Oracle Rdb7 version 7.0.1.2.

3.2.3 Transaction Changes in Stored Procedures Not Processed Correctly by SQL Clients

Bugs 440469 and 433034.

In prior releases of Oracle Rdb7, the following problems existed related to transaction changes performed in stored procedures:

- The GET DIAGNOSTICS options TRANSACTIONS_COMMITTED and TRANSACTIONS_ROLLED_BACK would not account for COMMIT or ROLLBACK statements issued from within other nested stored procedures. For example, the stored procedure C_TXN commits a transaction, but the GET DIAGNOSTICS in the calling routine incorrectly shows that zero transactions were committed.

  ```
  SQL> SET TRANSACTION READ WRITE;
  SQL> BEGIN
  cont> DECLARE :S0, :S1, :S2 INTEGER;
  cont> CALL C_TXN ();
  cont> GET DIAGNOSTICS
  cont>   :S0 = TRANSACTION_ACTIVE,
  cont>   :S1 = TRANSACTIONS_COMMITTED,
  cont>   :S2 = TRANSACTIONS_ROLLED_BACK;
  cont> TRACE :S0, :S1, :S2;
  cont> END;
  ~Xt: 0          0          0
  SQL>
  ```

- When a transaction was started or terminated within a nested stored procedure, the client interface was not made aware of the change in the transaction state. These state changes occurred when a transaction was started implicitly by a statement in a procedure or function, explicitly started with SET TRANSACTION statement, or terminated by a COMMIT or ROLLBACK statement.

  This might result in unexpected errors such as %RDB-E-BAD_TRANS_ HANDL, invalid transaction handle

- Extra rollback in multistatement procedure.

When a multistatement or stored procedure did a COMMIT or ROLLBACK and started a new transaction, the next use of the procedure would erroneously rollback the new transaction. Then, when the procedure did something requiring a transaction to be active, an error, BAD_TRANS_ HANDL, would result.

- When a ROLLBACK occurred in a stored procedure, SQL did not correctly close the WITH HOLD PRESERVE ON COMMIT or WITH HOLD PRESERVE NONE (the default) cursors.
- When a COMMIT occurred in a stored procedure, SQL did not correctly close the WITH HOLD PRESERVE ON ROLLBACK or WITH HOLD PRESERVE NONE (the default) cursors.
- When a HOLD cursor was active, it was possible that SQL would close the cursor when the cursor specified that it remain open across ROLLBACK or COMMIT statements. This occurred when a SET TRANSACTION statement was performed within a stored procedure.

These problems have been corrected in Oracle Rdb7 Release 7.0.1.2.

3.2.4 CREATE TABLE ... COLUMN COMPUTED BY Using a Variable Created Bugcheck Error

Bug 371841.

In prior versions of Oracle Rdb, a bugcheck error would result if a column of a table was defined using a COMPUTED BY clause that referenced a previously defined variable.

The following example shows this problem:

```
SQL> DECLARE :X INT;
SQL> CREATE TABLE VART (A INT, B COMPUTED BY (:X));
%SQL-I-BUGCHKDMP, generating bugcheck dump file USER5:[SMITH]SQLBUGCHK.DMP;
%SQL-F-BUGCHK, There has been a fatal error. Please contact your Oracle support
representative. SQL$BLRXPR - 15
```

This problem has been fixed in Oracle Rdb7 Release 7.0.1.2. It is still illegal to use a variable in this manner but, instead of the bugcheck error, the user will get an error message such as that shown in the following example.

```
SQL> DECLARE :X INT;
SQL> CREATE TABLE VART (A INT, B COMPUTED BY (:X));
%SQL-F-INVVARREF, Variable X is illegal in this context
```

3.2.5 Inserting Values that Contain a SELECT Statement and COALESCE Function Produced an Error

Inserting values that included a SELECT statement with a COALESCE function could result in a %RDB-E-INVALID_BLR error.

The following example demonstrates this problem:

```
SQL> CREATE DATA FILE FOO;
SQL> CREATE TABLE T1 (C1 SMALLINT, C2 CHAR(2), C10 CHAR(1), C11 INTEGER);
SQL> CREATE TABLE T3 (C1 SMALLINT, C2 CHAR(2), C3 INT, C4 INT);
SQL> INSERT INTO T3 (SELECT C1,C2,
cont>     (SELECT COALESCE(SUM(S2.C11),0) FROM T1 S2 ),
cont>     (SELECT COALESCE(SUM(S2.C11),0) FROM T1 S2 )
cont>     FROM T1 S GROUP BY C1,C2);
%RDB-E-INVALID_BLR, request BLR is incorrect at offset 174
SQL>
```

There are no simple workarounds although you can use intermediate variables in a multistatement procedure or 3GL program to hold the values that will be stored.

This problem has been corrected in Oracle Rdb7 Version 7.0.1.2.

3.2.6 CREATE MODULE Command Could Fail with Exceeded Quota (EXQUOTA) Error

Bug 588403.

In prior versions of Rdb, it was possible, in rare cases, for the CREATE MODULE command to exhaust all available virtual memory and fail with an "exceeded quota" error as shown in the example below.

```
SQL>  CREATE MODULE MODTST
cont>  LANGUAGE SQL
cont>  PROCEDURE TST();
cont>  BEGIN
cont>  DELETE FROM TEMP;
cont>  INSERT INTO TEMP (N, STRA, STRB)
cont>  SELECT
cont>    N,
cont>    T1.SA,
cont>    T2.SB
cont>  FROM
cont>    C
cont>  NATURAL LEFT OUTER JOIN
cont>    (SELECT N, SA FROM A) T1
cont>  NATURAL LEFT OUTER JOIN
cont>    (SELECT N, SB FROM B) T2;
cont>  END;
cont> END MODULE;
%COSI-F-EXQUOTA, exceeded quota
-SYSTEM-F-EXQUOTA, process quota exceeded
```

This occurred when a stored procedure used a NATURAL join clause in a SELECT statement in the procedure body. The join must have been between a base table (or view) and a derived table. The error occurred in the collection of language semantics information for the natural join. It is unlikely that this failure would occur when using a natural join between base tables or views.

The only workaround to this problem is to replace the natural join with equivalent inner or outer joins using ON or USING clauses.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.2.

3.2.7 External Functions Which Perform SQL Commands Could Return Incorrect SQLCODE/SQLSTATE Values

Bug 469633.

In previous versions of Rdb, when executing an external function that performed SQL commands, incorrect SQLCODE/SQLSTATE values could be returned. This problem did not affect external functions that did not perform SQL commands.

For example, the incorrect SQLCODE value indicating success could be returned to the query which referenced the external function when it should really be end-of-stream (SQLCODE_EOS(100)). The SQLSTATE value was then derived from this value and would also be incorrect. This usually resulted in the application interpreting the returned data incorrectly.

The problem was that the SQL context was reset during the external function execution and not correctly restored upon return to the caller. There is no workaround for this problem.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.2.

3.2.8 Memory Leak on Database Attach and Disconnect

Bug 536932.

A memory leak was observed in the SQL interfaces when doing database attaches and disconnects for previous versions of Oracle Rdb.

The following code example shows the problem when SQL memory tracing was turned on:

```
#include <stdio.h>
#include <ssdef.h>
#include <stdlib.h>
#include "sys$library:sql_literals.h"
int  sql$signal( void );
int  main( void )
{
        static int   i;
        exec sql include sqlca;
        EXEC SQL WHENEVER SQLWARNING GOTO error_label;
        EXEC SQL WHENEVER SQLERROR   GOTO error_label;
        /*  trace virtual memory calls in SQL$SHR.EXE */
        for ( i = 1; i <= 10; i ++ ) {
               exec sql attach 'filename mf_personnel';
               exec sql disconnect default;
        }
      exit( EXIT_SUCCESS );
error_label:
      sql$signal();
      exit( EXIT_FAILURE );
}
```

There is no workaround for this problem.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.2.

3.2.9 SQL Calculated Incorrect Character Length

In SQL, the length of an object of CHAR (or VARCHAR) datatype cannot exceed 65535 octets (bytes). If the character set for the database and/or the user session is DEC_MCS (which is the default), the maximum number of allowable characters for an object is also 65535 because only one octet is required to represent a DEC_MCS character.

However, if a multi-byte character set is specified for the database (such as DEC_Kanji requiring two octets to represent each character), the maximum allowable number of characters for the object will be reduced accordingly because less characters can be represented by the fixed 65535 octets.

When SQL checked if the length of a character string exceeded the maximum limit, it took no account of the multi-byte character set issue. This resulted in incorrect character length calculation.

This problem has been fixed in Oracle Rdb7 Release 7.0.1.2.

3.2.10 Unexpected UNSFIXINT Error from SUBSTRING Function

Bug 540404.

In Oracle Rdb7 Release 7.0.1, a problem could be encountered when using a CASE, DECODE, or COALESCE expression as part of a SUBSTRING function. When one of these conditional expressions was used in a FOR or FROM clause, SQL would incorrectly report that the expression produced a datatype other than the fixed numeric type required for this function. The following example shows the problem:

```
SQL> CREATE TABLE TEST_TABLE (COL1 CHAR(10));
SQL> INSERT INTO TEST_TABLE VALUES ('100.0');
1 row inserted
SQL> SELECT SUBSTRING (COL1
cont>     FROM 1
cont>     FOR CASE
cont>         WHEN POSITION('.' IN COL1) = 0
cont>         THEN CHAR_LENGTH(COL1)
cont>         ELSE (POSITION('.' IN COL1) - 1)
cont>         END)
cont>  FROM TEST_TABLE;
%SQL-F-UNSFIXINT, SUBSTRING must specify an unscaled fixed numeric
```

A workaround for this problem is to enclose the conditional expression in a CAST function which converts the result to an INTEGER.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.2.

3.2.11 Query Header Inherited for Derived Table Columns in Interactive SQL

When performing an interactive query from a derived table, SQL would inherit the query header defined for any column selected. This query header could be overridden using the AS clause to rename a column.

This first example shows a simple table with no query header which is displayed with the name specified in the derived table column list.

```
SQL> CREATE TABLE Q (A INTEGER);
SQL> INSERT INTO Q VALUES (1);
1 row inserted
SQL> INSERT INTO Q VALUES (2);
1 row inserted
SQL> SELECT * FROM (SELECT A FROM Q) AS TQ (NAME);
        NAME                                    -- derived column name
           1
           2
2 rows selected
SQL> ROLLBACK;
```

This second example shows how the query header is used for the column heading.

```
SQL> CREATE TABLE Q (A INTEGER QUERY HEADER IS 'A');
SQL> INSERT INTO Q VALUES (1);
1 row inserted
SQL> INSERT INTO Q VALUES (2);
1 row inserted
SQL> SELECT * FROM (SELECT A FROM Q) AS TQ (NAME) WHERE NAME > 1;
           A                                    -- query header
           2
1 row selected
SQL> ROLLBACK;
```

This final example shows that the AS renaming clause can be used to override the query header. This problem is corrected in Oracle Rdb7 Release 7.0.1.2. In previous releases the query header was always used.

```
SQL> CREATE TABLE Q (A INTEGER QUERY HEADER IS 'A');
SQL> INSERT INTO Q VALUE (1);
1 row inserted
SQL> INSERT INTO Q VALUE (2);
1 row inserted
SQL> SELECT AA AS NEW_NAME FROM (SELECT A FROM Q) AS QQ (AA) WHERE AA > 1;
    NEW_NAME                                    -- renamed column
           2
1 row selected
SQL> ROLLBACK;
```

3.2.12 Data In Temporary Tables Not Properly Deleted when Using SQL/Services

Bug 550873.

When using temporary tables with SQL/Services, the data in the temporary table was not properly deleted at commit time, even though the table attribute ON COMMIT DELETE ROWS was specified. When temporary tables are created, ON COMMIT DELETE ROWS is the default option, if not specified.

A workaround to this problem is to explicitly delete the data in the temporary table and not rely on the ON COMMIT DELETE ROWS option when using SQL/Services with temporary tables.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.2.

3.2.13 Problems with Builtin Functions COALESCE and NVL Datatypes

In earlier versions of Oracle Rdb7, users may have noticed SQL-F-DATTYPUNK errors or missing data when using COALESCE and NVL with parameter markers (or bind variables in Oracle7 terms).

The following example shows a SQL statement using NVL with a parameter marker as an argument. This statement would have produced the %SQL-F-DATTYPUNK error when the statement was prepared.

```
SQL> UPDATE EMPLOYEES
cont>    SET FIRST_NAME = NVL(?, '     ')
cont>    WHERE EMPLOYEE_ID = '00001';
```

This problem can be worked around by wrapping the host variable or parameter marker with a call to a builtin function that will not alter the argument's value and that will return a known type. For example, a character string argument could be wrapped with a call to LTRIM.

```
SQL> UPDATE EMPLOYEES
cont>    SET FIRST_NAME = NVL(LTRIM(?, '.'), '     ')
cont>    WHERE EMPLOYEE_ID = '00001';
```

The second argument in the call to LTRIM is any character literal that is known to not occur in any value that will be given to the parameter marker, so that the value given to the parameter marker will not be altered by the call to LTRIM. In this example, '.' was used.

These problems have been corrected and Oracle Rdb7 Release 7.0.1.2. In some instances, where SQL cannot determine a proper datatype for a parameter marker, it will describe the variable as being a VARCHAR(2000). This is most commonly seen with the INSERT statement.

3.2.14 Unexpected Errors After a CREATE MODULE Statement Failed

In prior releases of Oracle Rdb7, a failure during a CREATE MODULE statement might cause unexpected errors from subsequent CREATE MODULE and DROP MODULE statements which reference the name of the failing module. This is shown in the following example.

```
SQL> ATTACH 'FILENAME DB$:SCRATCH';
SQL> SET FLAGS 'TRACE';
SQL>
SQL> -- This create statement will fail because of the assignment to a constant
SQL> CREATE MODULE SAMPLE
cont>   LANGUAGE SQL
cont>     PROCEDURE P2 (IN :A INTEGER);
cont>       BEGIN
cont>       DECLARE :X CONSTANT INTEGER = 0;
cont>       SET :X = :A;
cont>       END;
cont> END MODULE;
%RDB-E-NO_META_UPDATE, metadata update failed
-RDB-E-INVALID_BLR, request BLR is incorrect at offset 79
-RDMS-E-READONLYVAR, variable (1) has been marked as CONSTANT and may not be updated
SQL>
SQL> CREATE MODULE SAMPLE
cont>   LANGUAGE SQL
cont>     PROCEDURE P1 (IN :A INTEGER);
cont>     BEGIN
cont>     DECLARE :X INTEGER;
cont>     SET :X = :A;
cont>     END;
cont> END MODULE;
%RDB-E-NO_META_UPDATE, metadata update failed
-RDMS-E-MODEXTS, there is another module named SAMPLE in this database
SQL>
SQL> DROP MODULE SAMPLE;
%SQL-F-MODNOTDEF, module SAMPLE is not defined
```

The second CREATE MODULE statement reported that the module already exists because it checked for loaded modules (which may be stored or non-stored). This test was made against the cached memory version of the system tables' metadata. The error resulted because the failing module was still partially resident in memory. A workaround is to DISCONNECT from the database after such a failure.

The DROP MODULE statement was executed in an attempt to cleanup the module which the CREATE MODULE statement indicated still existed. This command referenced the on-disk metadata to validate the name and found no matching module name because the error removed all references.

Note

The example has been contrived to show the symptoms of this problem. There are many different types of errors which could be detected at runtime which would leave the module in this state.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.2. The partially loaded module information is now purged when errors occur during the CREATE MODULE statement.

3.2.15 Unexpected CSETBADASSIGN Error when Function Returned a Character Varying Datatype

In prior releases of Oracle Rdb7, stored and external functions could not be used if they returned VARCHAR, LONG VARCHAR or NATIONAL CHARACTER VARYING datatypes and the CHARACTER SET was not compatible with MCS (Multinational Character Set). The following example shows the problem:

```
SQL> ATTACH 'FILENAME DB$:SCRATCH';
SQL> SET FLAGS 'TRACE';
SQL> SET DEFAULT CHARACTER SET 'DEC_KANJI';
SQL> SET NATIONAL CHARACTER SET 'DEC_KANJI';
SQL> SET IDENTIFIER CHARACTER SET 'DEC_KANJI';
SQL> SET LITERAL CHARACTER SET 'DEC_KANJI';
SQL> SET CHARACTER LENGTH 'CHARACTERS';
SQL>
SQL> CREATE MODULE SAMPLE
cont>   LANGUAGE sql
cont>     FUNCTION P1 (IN :A INTEGER)
cont>      RETURNS LONG VARCHAR;
cont>      RETURN CAST(:A AS LONG VARCHAR);
cont>
cont>     PROCEDURE P2 (IN :A INTEGER);
cont>       BEGIN
cont>       DECLARE :X LONG VARCHAR;
cont>       SET :X = P1 (:A) || P1 (:A);
cont>       END;
cont> END MODULE;
%RDB-E-NO_META_UPDATE, metadata update failed
-RDB-E-CONVERT_ERROR, invalid or unsupported data conversion
-RDMS-E-CSETBADASSIGN, incompatible character sets prohibit the requested assignment
```

This problem has been corrected in Oracle Rdb7 Release 7.0.1.2. Functions can now return any character set for these datatypes. In prior versions of Rdb7, the datatype was defaulting to MCS at runtime. The existing definitions for external and stored functions are correct and need not be changed for Oracle Rdb7 Release 7.0.1.2.

3.2.16 Unexpected Value on Date/Time Arithmetic Overflow

In prior releases of Oracle Rdb7, if an overflow occurred during execution of date /time arithmetic, the result was displayed as a zero value, rather than raising an exception. The following example shows the problem.

```
SQL> SELECT INTERVAL '3649634:23:59' DAY(7) TO MINUTE
cont>   - INTERVAL '-00500057:02:32' DAY(8) TO MINUTE
cont> FROM AAA LIMIT TO 1 ROW;

  -00000000:00:00
1 row selected
```

This problem has been corrected in Oracle Rdb7 Release 7.0.1.2. If an overflow occurs during the execution of date/time arithmetic, the exception will be raised and displayed as shown in the following example.

```
SQL>  SELECT INTERVAL '3649634:23:59' DAY(7) TO MINUTE
cont>    - INTERVAL '-00500057:02:32' DAY(8) TO MINUTE
cont> FROM AAA LIMIT TO 1 ROW;
%RDB-E-ARITH_EXCEPT, truncation of a numeric value at runtime
-COSI-F-IVTIME, invalid date or time
```

3.2.17 Default Value Added with ALTER DOMAIN Statement Could Be Incorrect

Bug 456353.

In prior versions of Oracle Rdb, a DATE VMS default value could be added with the origin date, 17-NOV-1858, by using a default value containing a string of zero digits ('0'). The same default value added with ALTER DOMAIN would create a random time value instead of the expected 00:00:00.00.

This method was not documented and was really a side effect of defining an illegal date value. This method has now been made consistent with CREATE and ALTER commands in Oracle Rdb7 Release 7.0.1.2.

Oracle recommends that the origin timestamp '17-NOV-1858 00:00:00.00' be used instead of relying on this method in the future. Default values which used this date format did not have the reported problem.

3.2.18 Unexpected Bugcheck Error in Routine RDMS$$PRIV_CHECK_ACCESS

Bug 555596.

In prior releases of Oracle Rdb7 it was possible, under special circumstances, to generate a bugcheck error in routine RDMS$$PRIV_CHECK_ACCESS at offset 000002D8. The circumstances were:

- Oracle Rdb7 was running on OpenVMS for Alpha.
- A CREATE INDEX statement failed for reasons such as the detection of a duplicate row for a UNIQUE index.
- The transaction was immediately restarted using a SET TRANSACTION statement which contained a RESERVING clause.

The table specified in the reserving clause was checked to ensure that the current user had READ access. It was during this check that stale information left over from the failing CREATE INDEX statement caused the bugcheck error in Oracle Rdb.

The workaround to this problem is to execute any simple query after the failing CREATE INDEX statement such as a SHOW INDEX statement. These statements will clear the stale information prior to the SET TRANSACTION statement.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.2.

3.3 Oracle RMU Errors Fixed

3.3.1 RMU/SHOW STATISTICS Bugcheck Error at KUTDIS$UPDATE_RS_ENT

Starting with Oracle Rdb7 Release 7.0.1.1, with certain combinations of numbers of logical areas and numbers of physical areas, the RMU/SHOW STATISTICS utility could fail with an access violation or a bugcheck error with an exception in the KUTDIS$UPDATE_RS_ENT routine.

For example, the following test case failed with an access violation:

```
$ RMU/CLOSE MF_PERSONNEL
$ MCR SQL$
SQL> ALTER DATABASE FILENAME MF_PERSONNEL
cont> RESERVE 1200 STORAGE AREAS;
%RDMS-W-DOFULLBCK, full database backup should be done to ensure
future recovery
SQL> EXIT;
$ RMU/OPEN MF_PERSONNEL
$ RMU/SHOW STAT MF_PERSONNEL

%SYSTEM-F-ACCVIO, access violation, reason mask=00,
virtual address=0000000000000000, PC=000000000044B834, PS=0000001B
%RMU-F-FATALOSI, Fatal error from the Operating System Interface.
%RMU-I-BUGCHKDMP, generating bugcheck dump file DKA0:[USER]RMUBUGCHK.DMP;
%RMU-F-FTL_SHOW, Fatal error for SHOW operation at 28-JAN-1998 12:42:02.79
```

There is no known workaround for this problem beyond avoiding the use of the RMU/SHOW STATISTICS utility.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.2.

3.3.2 Bugchecks After Conversion to Oracle Rdb7 Release 7.0

Bug 612051.

After converting a database to Oracle Rdb7 Release 7.0 with the RMU/CONVERT command, it was possible to generate bugcheck errors at exceptions similar to PIOFETCH$WITHIN_DB and DIOFETCH$FETCH_SNAP_SEG.

This problem was due to incorrectly initialized transaction sequence blocks (TSNBLK). TSNBLKs are rootfile data structures that contain information about transactions including their transaction sequence number (TSN).

Because the size of the TSN changed from 32-bit to 64-bit in V7.0, less TSNs can be represented in one TSNBLK. Depending on the number of users and nodes allocated in your database, the initial number of TSNBLKs may need to be increased during the convert process. A problem was discovered where these new TSNBLKs were not correctly initialized during allocation and their contents are undetermined.

To determine if you may be affected by this problem, you can use the RMU/DUMP /HEADER/OPT=DEBUG command to dump the contents of your rootfile. You can then search the output for "TSNBLK_ENT" to see if some of the latter TSNBLKs appear to be uninitialized (very large TSN values appear).

It is important to note that this problem does not occur if you implicitly converted your database using the RMU/RESTORE command to restore from a prior version's backup file.

As a workaround, perform the conversion with the RMU/CONVERT/NOCOMMIT command. You can then issue the RMU/CONVERT/ROLLBACK command and do another convert. This sequence will properly initialize the TSNBLKs in the Oracle Rdb7 rootfile.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.2.

3.3.3 RMU/VERIFY/INDEX/CHECKSUM_ONLY Incorrectly Reported a BADIDXREL Error

Bug 608002.

Performing an RMU/VERIFY command using both the /CHECKSUM_ONLY and /INDEX qualifiers resulted in one or more spurious BADIDXREL error messages being reported. The code incorrectly reported that an index node was pointing to a non-existent data record when, in fact, the data record did exist.

Beginning with Oracle Rdb7, the RMU VERIFY command used a new method to verify indexes. In prior versions, the verify operation tried to retrieve the table row to which the index pointed. Beginning with Oracle Rdb7, the verify operation created a sorted list of all dbkeys for a table and a sorted list of all dbkeys in an index. By comparing these two lists, the verify operation could detect any cases of an index missing an entry for a data row. The table record sort list was built during the page segment verification phase while the index sort list was built during the index verification phase. At a subsequent point in the processing, the RMU VERIFY command performed its index/data verification where it compared the two lists to detect inconsistencies. Note that, when the /INCREMENTAL qualifier is used, RMU reverted to the method of index verification used in prior versions of Oracle Rdb. This is because the full set of index and table dbkeys could not be processed when incrementally verifying a database.

The spurious BADIDXREL problem was caused by the fact that, when the /CHECKSUM_ONLY qualifier was used, the RMU VERIFY command skipped performing the normal page segment verification step and it was in this step that the dbkeys for the data records were placed into the table sort list. By the time the index/data verification step started, there was a full list of index dbkeys and an empty list of table record dbkeys. This caused the BADIDXREL message(s) to be generated. To solve this problem, RMU reverted to the method of index verification used in prior versions of Oracle Rdb when the /CHECKSUM_ONLY qualifier is present, just as is done when using the /INCREMENTAL qualifier.

The following example demonstrates the problem:

```
$ RMU/VERIFY/INDEX=DEPARTMENTS_INDEX/AREA=DEPARTMENTS/CHECKSUM_ONLY MF_PERSONNEL
%RMU-W-BADIDXREL, Index DEPARTMENTS_INDEX either points to a non-existent record or
has multiple pointers to a record in table DEPARTMENTS.
The logical dbkey in the index is 64:2:1.
```

As a workaround, it is possible to perform the database verification in two steps:

1. Do a CHECKSUM_ONLY verification without index verification.
2. Do an INDEX verification with full page segment verification.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.2.

3.3.4 RMU/REPLICATE AFTER REOPEN_LOG Created Logfile with No Contents

The RMU/REPLICATE AFTER_JOURNAL REOPEN_OUTPUT command opens log files on the master and standby servers when the hot standby feature is used. The standby log file was updated correctly but the master log file was not getting any information written to it.

The following example shows that only one line of information was written to the log file on the master server.

```
9-JAN-1998 14:47:38.05 - Sending LCS_REOPEN_LOG (1:0)
```

There is no workaround to this problem.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.2. The database monitor has been corrected to notify the various hot standby servers properly when subsequent output files should be re-opened.

3.3.5 RMU/SHOW STATISTICS "Logical Area" Statistics Excluded Ranked B-tree Indexes

The RMU/SHOW STATISTICS utility did not display any statistics in the "Logical Area" screen when ranked B-tree indexes were selected.

There is no workaround to this problem.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.2. Ranked B-tree index statistics are now displayed in the "Logical Area" screen.

3.3.6 RMU/SHOW STATISTICS "Lock Timeout Logfile" Did Not Contain Any Messages

The RMU/SHOW STATISTICS utility did not report any lock timeout information to the lock timeout logfile. This problem occurred when using the /LOCK_ TIMEOUT_LOG qualifier. The lock timeout was displayed on the "Lock Timeout History" screen correctly.

The following example shows how to produce a lock timeout.

In one window, issue the following RMU command:

```
$ RMU/SHOW STATISTICS/NOHISTORY /TIME=1 /NOINTERACTIVE -
/LOCK_TIMEOUT_LOG=TIMEOUT.LOG /NOBROADCAST /UNTIL="17:00" PERS
```

In another window, start a SQL session that exclusively locks a table:

```
$ SQL
SQL> ATTACH 'FILENAME PERSONNEL';
SQL> SET TRANSACTION RESERVING EMPLOYEES FOR EXCLUSIVE WRITE;
```

In a third window, start a SQL session that attempts to access the locked table:

```
$ SQL
SQL> ATTACH 'FILENAME PERSONNEL';
SQL> SET TRANSACTION RESERVING EMPLOYEES FOR SHARED WRITE WAIT ;
%RDB-E-LOCK_CONFLICT, request failed due to locked resource
-RDMS-F-TIMEOUT, timeout on logical area
```

The resulting log file contained only the header information with no timeout information.

```
Rate: 1.00 Second             Lock Timeout History          Elapsed: 00:04:02.84
Page: 1 of 1      DISK$USER1:[DB.V70]MF_PERSONNEL.RDB;1     Mode: Online

Process.ID Occurred...   Lock.timeout.reason..................... #Timeout
3D400349:1 08:27:36.65 - waiting for logical area 85 (CW)                 3
3D40035A:1 08:25:53.34 - waiting for logical area 85 (CW)                 1
```

There is no workaround to this problem.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.2. The lock timeout information is now written to the logfile correctly as shown in the following example:

```
 Oracle Rdb X7.0-00 Performance Monitor Lock Timeout Log
 Database DISK$:[USER]PERS.RDB;1
 Lock Timeout Log created 26-SEP-1997 14:16:02.60
2AA0361C:1  14:40:03.16 - waiting for logical area 56 (CW) [2 missed]
```

3.3.7 RMU/SHOW STATISTICS "User-Defined Events" Did Not Work for "Stored Snap Record" Field

A user-defined event could not be created on any of the "Snapshot Statistics" screen fields using the RMU/SHOW STATISTICS utility. The event was rejected because the particular statistics field could not be found. However, importing the same configuration file worked correctly.

Only the "Snapshot Statistics" screen was affected by this problem. All other screens worked correctly.

The following configuration file entry is an example where the identified statistics field could not be defined:

```
EVENT_DESCRIPTION="ENABLE 'stored snap record' MAX_CUR_TOTAL INITIAL 200
EVERY 100 LIMIT 50 INVOKE DB_ALERT";
```

The entry above produced the following error in the log file:

```
   .
   .
   .
line 66: variable "EVENT_DESCRIPTION" value "ENABLE 'stored snap record'
MAX_CUR_TOTAL INITIAL 200 EVERY 100 LIMIT 50 INVOKE DB_ALERT"
line 66: event statistic field "stored snap record" not found
```

A workaround is to "import" the configuration file once the displays are available.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.2. The RMU/SHOW STATISTICS utility now supports the definition of user-defined events on all statistic screens, including the "Snapshot Statistics" screen.

3.3.8 RMU/SHOW STATISTICS "Stall Messages" Contained Unusual Stall Messages

When using the RMU/SHOW STATISTICS "Stall Messages" screen during cluster statistics collection, it was sometimes possible to have unusual stall messages displayed.

The problem occurred when more than 96 users were attached to the database, on any node of the cluster.

The following example shows some of the unusual messages:

```
Cluster: SDVE01 (3/3/16)Oracle Rdb V7.0-1 Perf. Monitor  12-DEC-1997 13:43:38.64
Rate: 1.00 Second               Stall Messages            Elapsed: 00:46:41.60
Page: 1 of 1               USER1:[DB]SGARDB.RDB;1         Mode: Online
------------------------------------------------------------------------------
Process.ID Since..... T Stall.reason............................ Lock.ID.
204304AE:1s00:00:00.00 W
20805974:1*00:00:00.00 W
20449D3E:3*00:00:00.00 -
20209581:1s13:43:38.64 - performing remote statistics collection
208058C0:1s00:00:00.00 W Message number F2C8FFFF                   FFFFFFFF
2046CB23:3*00:00:00.00 W Message number F2C8FFFF                   FFFFFFFF
2080596C:2*00:00:00.00 W Message number F2C8FFFF                   FFFFFFFF
2044B480:3*00:00:00.00 W Message number F2C8FFFF                   FFFFFFFF
20450CCA:2*00:00:00.00 W Message number F2C8FFFF                   FFFFFFFF
20444127:3*00:00:00.00 - Message number F2C8FFFF                   FFFFFFFF
2042164B:3*00:00:00.00 W Message number F2C8FFFF                   FFFFFFFF

------------------------------------------------------------------------------
```

There is no workaround to this problem.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.2. The stall messages are displayed properly during cluster statistic collection operations.

3.3.9 Parallel Load Without Power Utilities Created Incorrect Error

Bug 431754.

Performing an RMU/LOAD command using the /PARALLEL qualifier resulted in a confusing error message if the Power Utilities option was not installed. It is required that the Power Utilities option be installed in order to perform a parallel load operation. Because the RMU/LOAD command was not properly checking for the power utilities option, a confusing error message was displayed when this option was not installed, as the following example demonstrates:

```
$ RMU/LOAD/RECORD_DEFINITION=(FILE=EMP.RRD)/PARALLEL MF_PERSONNEL EMPTAB EMP.UNL
%RMU-F-UNEXPEXECTERM, Unexpected termination by executor EXECUTOR_1 (exit code
= 98962.)
%RMU-I-DATRECREAD,  200 data records read from input file.

%RMU-I-EXECSTAT0, Statistics for EXECUTOR_1:
%RMU-I-EXECSTAT1,   Elapsed time:  00:00:00.00    CPU time:       0.0
%RMU-I-EXECSTAT2,   Storing time:  00:00:00.00    Rows stored:    0
%RMU-I-EXECSTAT3,   Commit time:   00:00:00.00    Direct I/O:     0
%RMU-I-EXECSTAT4,   Idle time:     00:00:00.00    Early commits:  0

%RMU-I-EXECSTAT5,   Main process idle time:  00:00:00.00
%RMU-I-DATRECSTO,   0 data records stored.
```

The following example shows the error message displayed when RMU Load correctly checks for the Power Utilities option before attempting the parallel load operation:

```
$ RMU/LOAD/RECORD_DEFINITION=(FILE=EMP.RRD)/PARALLEL MF_PERSONNEL EMPTAB EMP.UNL
%RMU-F-FILACCERR, error searching for file SYS$LIBRARY:RDMRLE.EXE
-RMU-E-CHKPOWUTL, Make sure that the Power Utilities option has been properly
installed on your system
%RMU-F-FTL_LOAD, Fatal error for LOAD operation at 30-DEC-1997 08:04:44.16
```

This problem has been corrected in Oracle Rdb7 Release 7.0.1.2.

3.3.10 Erroneous RMU$_DENSITY Errors

Bug 424441.

All OpenVMS platforms.

When /DENSITY=0 was specified on a RMU/BACKUP command, it indicated that a tape drive's default density will be used. After RMU set the drive's density, it would then read the density back. But some tape drives did not report back zero if they were set to zero. They reported back their real default density and this made the comparison fail resulting in RMU$_DENSITY errors.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.2. The check of density if /DENSITY=0 was specified has been eliminated.

3.3.11 Operator Intervention Requested on Backup

Bug 490826.

All OpenVMS platforms.

When backing up to some TZ model tape drives with automatic tape loaders, if the tape mechanism was slightly sluggish, the request for a second tape volume after the first could take just long enough that the request timed out and operator intervention is called for.

This problem occurred in previous versions of Oracle RMU.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.2. The timeout for mount requests for most types of tape drives has been increased to 225 seconds.

3.3.12 RMU/COLLECT OPTIMIZER_STATISTICS Assigns Zero Cardinalities for Some Tables

Bug 507724.

In prior releases of Oracle Rdb7, the RMU/COLLECT OPTIMIZER_STATISTICS command would assign zero cardinalities for tables and indexes which were mapped, by default, to a storage area other than RDB$SYSTEM.

If a database was created or imported with the clause DEFAULT STORAGE AREA that referenced an area other than RDB$SYSTEM, then the RMU/COLLECT utility was unable to determine the correct cardinalities. This was because this utility incorrectly assumed unmapped tables resided in the RDB$SYSTEM storage area, as was the case in older versions of Oracle Rdb.

Workarounds for this problem include using the RMU/ANALYZE/CARDINALITY /UPDATE command to set the cardinalities for the tables, or recreating the database using the DEFAULT STORAGE AREA RDB$SYSTEM clause.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.2. The RMU/COLLECT OPTIMIZER_STATISTICS utility now correctly locates and processes tables and indexes in the default storage area. The RMU/COLLECT command should be run on any database which was created with the DEFAULT STORAGE AREA clause so that the cardinalities are correctly stored.

3.3.13 RMU/ANALYZE/INDEX Sorted Ranked Index RMU$FLAGS Anomaly

Bug 425320.

Performing a RMU/ANALYZE/INDEXES command on a sorted ranked index using the /BINARY_OUTPUT qualifier resulted in an incorrect value being assigned to the RMU$FLAGS field. For summary records, the RMU analyze indexes command assigned the RMU$FLAGS field a value of 4 for a non-unique sorted ranked index and a value of 5 for a unique sorted ranked index. This was done to distinguish it from the values assigned to RMU$FLAGS for non-unique and unique sorted nonranked indexes, which is 0 and 1, respectively. However, this caused a conflict because non-summary records which were generated for each index node level for non-unique and unique sorted nonranked indexes already used the values 4 and 5.

Therefore, the RMU/ANALYZE/INDEX command was changed to assign the following 12 possible values to the RMU$FLAGS field:

- 0 - Index is sorted and not unique. A full report is not generated.
- 1 - Index is sorted and unique. A full report is not generated.
- 2 - Index is hashed and not unique. A full report is not generated.
- 3 - Index is hashed and unique. A full report is not generated.
- 4 - Index is sorted and not unique. A full report is generated.
- 5 - Index is sorted and unique. A full report is generated.
- 6 - Index is hashed and not unique. A full report is generated.
- 7 - Index is hashed and unique. A full report is generated.

- 8 - Index is sorted ranked and not unique. A full report is not generated.
- 9 - Index is sorted ranked and unique. A full report is not generated.
- 12 - Index is sorted ranked and not unique. A full report is generated.
- 13 - Index is sorted ranked and unique. A full report is generated.

The RMU/ANALYZE/INDEX command uses the RMU$FLAGS bits shown in Table 3–2 for describing specific index information.

Table 3–2 RMU$FLAGS Bits Used by the RMU/ANALYZE/INDEX Command

Bit Offset	Meaning
0	Unique index if true
1	Hashed index if true
2	Full report record if true
3	Ranked index if true

This problem was corrected in Oracle Rdb7 Release 7.0.1.2.

3.3.14 RMU/ANALYZE/INDEX Sorted Ranked Index Offset Anomaly

Bug 425320.

Performing a RMU/ANALYZE/INDEX command using the /BINARY_OUTPUT qualifier resulted in incorrect values assigned to all fields starting with RMU$DUPLICATE_USED through RMU$TOTAL_IKEY_COUNT. A problem was caused because a new field, RMU$DUPLICATE_MAP, was added before the RMU$DUPLICATE_USED field. However, the definition for this field was not being written to the output record definition file. This caused all field values starting at RMU$DUPLICATE_USED to use the wrong field offset when the binary data was loaded using the record definition file with RMU Load.

The RMU$DUPLICATE_MAP field contains the count of the number of duplicate bit maps for a sorted ranked index. In the case of other index types, this field will have a value of zero. The datatype for this field is F_FLOATING. The RMU$DUPLICATE_MAP field is now being written to the output record definition file.

This problem was corrected in Oracle Rdb7 Release 7.0.1.2.

3.3.15 RMU/BACKUP/AFTER_JOURNAL Stalled Following AIJ Backup Completion

Using the RMU/BACKUP/AFTER_JOURNAL utility, it was possible for the utility to stall following completion of an AIJ backup operation.

The problem was caused by a race condition (timing related) while trying to update the process-global symbols. The stall involved the AIJ backup utility waiting for the AIJ global lock.

The workaround is to use the AIJ backup server (ABS) instead of the manual RMU/BACKUP/AFTER_JOURNAL statement. The ABS server does not exhibit this problem.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.2. The RMU/BACKUP/AFTER_JOURNAL utility no longer waits for the AIJ global lock while trying to update the process-global symbols.

3.4 Hot Standby Errors Fixed

3.4.1 Hot Standby Bugcheck Error and Shutdown During Large Transaction Update

Bug 550394.

When using the hot standby feature, it was possible for an extremely large single-transaction update, or series of transaction updates to cause an AIJ group-commit buffer overflow. This overflow caused the log replication servers to create bugcheck errors and shutdown hot standby replication. Hot standby replication could not then be restarted because the expected AIJ journal could not be located.

The workaround is to define the RDM$BIND_AIJ_IO_MAX logical name in the LNM$SYSTEM_TABLE to the value "96" before opening the master database. This limits the size of the group-commit cache to a size which is reasonable for network communications. This size, in most cases, does not cause AIJ performance degradation.

Be sure to define this logical name on all nodes accessed by the master database.

It is vital that the logical name be defined prior to opening the master database. If the master database is already open, then use the RMU/SHOW STATISTICS utility "Dashboard" facility to change the size of the "Max IO Blocks" entry to "49152".

This problem has been corrected in Oracle Rdb7 Release 7.0.1.2. The AIJ group-commit buffer size has also been corrected when the hot standby feature is activated.

3.4.2 ALS Server Slow to Respond to Global Checkpoint Requests

When hot standby was being used, the AIJ Log Server (ALS) process was slow to respond to global checkpoint requests, typically issued by the AIJ Backup Server (ABS) process. This often resulted in the ABS process not being able to backup an AIJ journal within a reasonable timeframe.

The problem was further aggravated because repeated global checkpoint requests by the ABS often resulted in the ALS having to replicate checkpoint information from other attached processes, thereby making the ALS checkpoint information even more stale.

There is no workaround to this problem.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.2. The AIJ Log Server process now responds in a timely manner to global checkpoint requests.

3.4.3 Hot Standby LRS Server Started with Access Violation

When using the Hot Standby feature, it was sometimes possible for the replication server on the standby system to create a bugcheck error during startup with an access violation.

There is no workaround for this problem.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.2.

3.4.4 ALS Failure on a Node Could Cause Repeated AIJ Records on Another Node

Under rare circumstances, it was possible for the failure of an AIJ Log Server (ALS) to cause the last block written to the AIJ journal on another node to be repeated.

This problem only occurred when using the AIJ Log Server (ALS) processes. The problem required that a partial AIJ block be written on 1 node, then AIJ control passed to another node that uses the ALS. The ALS on the new node needed to write at least 1 block of AIJ journal information and then terminate prematurely. The original node must then regain control of the AIJ journal before the database recovery process (DBR) for the failed ALS was invoked by the database monitor. This was an extremely rare event.

The problem would not occur during normal ALS termination.

The problem could be detected by examining the AIJ journal. The dates of the repeated AIJ records would appear to be older than surrounding AIJ records. Also, there must have been a single "other" node between the repeated AIJ records.

The following example shows a case of the repeated AIJ records. Note that the records written to block 502548 by monitor ID 2 were repeated at block 534291. Also, all records between these 2 blocks were written by the same node, monitor ID 3 in this example. Notice that the date of the AIJ records written at block 534291 were 7 seconds earlier than the records at AIJ block 534290.

Note

Because of time variances within a cluster, you could not always rely on the AIJ record dates as a means of identifying this problem.

Also note that in this example, the repeated AIJ records were located 31,743 blocks away from each other!

```
502548/1115404   TYPE=G, LENGTH=16, TAD= 3-DEC-1997 14:18:33.04, CSM=C5
    Group commit date is  3-DEC-1997 14:18:33.04
    Message sequence number is 0
    Monitor ID is 2

502548/1115405   TYPE=D, LENGTH=424, TAD= 3-DEC-1997 14:18:33.04, CSM=00
    TID=2972, TSN=0:1177117, AIJBL_START_FLG=0, SEQUENCE=230
    Appending to partial AIJBL
        MODIFY: PDBK=15:1096:22, LDBID=173, PSN=155, FLAGS=00, ABM_PNO=2
                REC_LEN=144, LENGTH=144

502549/1115406   TYPE=G, LENGTH=16, TAD= 3-DEC-1997 14:18:35.36, CSM=C6
    Group commit date is  3-DEC-1997 14:18:35.36
    Message sequence number is 0
    Monitor ID is 3

502549/1115407   TYPE=D, LENGTH=492, TAD= 3-DEC-1997 14:18:35.36, CSM=00
    TID=8151, TSN=0:1177115, AIJBL_START_FLG=1, SEQUENCE=23
        MODIFY: PDBK=3:1560326:0, LDBID=0, PSN=6, FLAGS=00, LENGTH=38

.
. [remaining AIJ records from Monitor ID 3 removed for brevity]
.
```

```
534290/1185980   TYPE=G, LENGTH=16, TAD= 3-DEC-1997 14:25:15.13, CSM=C6
    Group commit date is  3-DEC-1997 14:25:15.13
    Message sequence number is 0
    Monitor ID is 3

534290/1185980   TYPE=V, LENGTH=229, TAD= 3-DEC-1997 14:25:15.13, CSM=00
    TSN=0:1177279

534291/1185981   TYPE=G, LENGTH=16, TAD= 3-DEC-1997 14:18:33.04, CSM=E1
    Group commit date is  3-DEC-1997 14:18:33.04
    Message sequence number is 0
    Monitor ID is 2

534291/1185982   TYPE=D, LENGTH=424, TAD= 3-DEC-1997 14:18:33.04, CSM=00
    TID=2972, TSN=0:1177117, AIJBL_START_FLG=0, SEQUENCE=230
    Partial AIJBL remains
```

Examination of the monitor log indicates that the ALS process on the monitor ID 3 node failed at 14:25:15 after writing the AIJ records at block 534290.

The best workaround is to disable the ALS process.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.2.

3.4.5 ALS Releasing Control to DBR on Same Node Could Corrupt AIJ File

It was possible for the AIJ Log Server process (ALS), when releasing control of the AIJ sub-system to a database recovery process (DBR) on the same node, to cause the DBR process to corrupt the AIJ file.

This problem occurred when the ALS finished formatting AIJ request blocks (ARBs) into a partial block as the last AIJ operation on the database, and the DBR requested control from the same node.

There is no workaround to this problem, other than ensuring that the DBR processes do not get invoked.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.2. The ALS process now invalidates the AIJ cache information when it releases control of the AIJ sub-system.

3.5 Row Cache Errors Fixed

3.5.1 The ALTER DATABASE ROW CACHE IS DISABLED Command Did Not Disable Logical Area Caches

Bug 468405.

Disabling row caching using the ALTER DATABASE ROW CACHE IS DISABLED command in interactive SQL did not disable logical area caching. Physical area caching was, however, disabled.

The workaround for this problem is to individually drop all row caches.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.2. Oracle Rdb now correctly disables the row cache feature after the ALTER DATABASE ROW CACHE IS DISABLED command has been used.

4

Software Errors Fixed in Oracle Rdb7 Release 7.0.1.3

This chapter describes software errors that were fixed by Oracle Rdb7 Release 7.0.1.3.

4.1 Software Errors Fixed That Apply to All Interfaces

4.1.1 A Query Using GROUP BY or DISTINCT with ORDER BY Returned Wrong Order

Bug 614647.

A SELECT statement with an explicit ORDER BY clause returned rows sorted in a different order than requested. The query must also have contained a GROUP BY or a DISTINCT clause and a subselect statement with a join order different from the ORDER BY clause.

The following query returned rows in the wrong order:

```
SQL> -- List each employee from Massachusetts and his/her average salary over all
SQL> -- jobs held.  The key factors in this problem are, in the order shown, the
SQL> -- sub-select statement with a match on the EMPLOYEE_ID column, a GROUP BY
SQL> -- clause, and an ORDER BY clause.
SQL>
SQL> SELECT
cont>    E.LAST_NAME,
cont>    E.FIRST_NAME,
cont>    'average salary is',
cont>    (SELECT AVG(SALARY_AMOUNT) FROM SALARY_HISTORY S2  -- <- key factor
cont>     WHERE
cont>         S2.EMPLOYEE_ID = S1.EMPLOYEE_ID AND     -- <- key factor
cont>         E.STATE       = 'MA')
cont> FROM
cont>    EMPLOYEES      E,
cont>    SALARY_HISTORY S1
cont> WHERE
cont>    E.STATE        = 'MA'    AND
cont>    S1.EMPLOYEE_ID = E.EMPLOYEE_ID
cont>  GROUP BY                              -- <- key factor
cont>    E.STATE,
cont>    E.LAST_NAME,
cont>    E.FIRST_NAME,
cont>    E.EMPLOYEE_ID,
cont>    S1.EMPLOYEE_ID
cont>  ORDER BY                              --<- key factor
cont>    E.LAST_NAME;
 E.LAST_NAME      E.FIRST_NAME
 Myotte           Daniel         average salary is    1.675080000000000E+004
 Siciliano        George         average salary is    1.225983333333333E+004
 Pfeiffer         Karen          average salary is    1.133525000000000E+004
 Gutierrez        Ernest         average salary is    2.799022222222222E+004
 Harrison         Lisa           average salary is    5.912966666666666E+004
```

```
 McElroy         Mary          average salary is    2.232966666666667E+004
 Rodrigo         Lisa          average salary is    1.198737500000000E+004
 Mistretta       Kathleen      average salary is    4.883144444444445E+004
 MacDonald       Johanna       average salary is    6.885655555555556E+004
9 rows selected
```

The results were sorted by EMPLOYEE_ID rather than by LAST_NAME.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.3.

4.1.2 Illegal Wildcard Usage Caused SQL Bugcheck Error

Bug 470328.

In prior releases of Oracle Rdb, a SQL statement which used a wildcard (*) column select expression in an incorrect context would generate a SQL bugcheck error as shown in the following example.

```
SQL> SELECT A.* || '|' FROM RDB$RELATIONS A;
%SQL-I-BUGCHKDMP, generating bugcheck dump file DISK:[DIR]SQLBUGCHK.DMP;
%SQL-F-BUGCHK, There has been a fatal error. Please submit a software
performance report. SQL$SEMASS - 9
```

The use of the wildcard (*) in this context is not allowed. Even if the table only contained one column, you must specify the column by name. The wildcard can only be used to select all the columns from a table

This problem has been corrected in Oracle Rdb7 Release 7.0.1.3. This illegal reference now generates an error as shown in the following example.

```
SQL> SELECT A.* || '|' FROM RDB$RELATIONS A;
%SQL-F-INVSELSTAR, * is not allowed in this context
```

4.1.3 Loss of Dbkeys from Ranked Index

Under rare conditions, an insertion of a Dbkey into a ranked index duplicate node could result in the loss of eight consecutive Dbkeys from the end of that duplicate node.

The loss of these Dbkeys would cause invalid Dbkey errors to be raised whenever an affected record was modified or erased. In addition, the results of a RMU/VERIFY command on the index would show that Dbkeys were missing from the index.

As a possible workaround for this problem, rebuild the existing index by dropping and recreating it or use a non-ranked index.

This problem has been corrected in Oracle Rdb7 Version 7.0.1.3.

4.1.4 Query Produced Bugcheck Error when Match Keys Were Different Datatypes

Bug 632149.

The following query produced a bugcheck error because the match keys were of different datatypes.

```
SELECT * FROM TA,TB WHERE CA=CB;
```

The table columns were defined for CHAR and VARCHAR datatypes.

```
SQL> CREATE TABLE TA (CA CHAR(5));
SQL> CREATE TABLE TB (CB VARCHAR(5));
```

The problem was caused because the zig-zag match strategy returned the wrong result if the match keys were of different datatypes. In this case, the query bugchecked when one of the match keys was a varying character datatype.

Change the datatype of VARCHAR to CHAR as a workaround.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.3.

4.1.5 RMU/VERIFY/INDEX Showed Cardinality Errors for Ranked Index

Bug 675923.

Under rare circumstances, the CREATE INDEX ... TYPE IS SORTED RANKED command could build an incorrect ranked index.

The following example demonstrates this problem:

```
SQL> ATTACH 'FILENAME TESTDB';
SQL> CREATE INDEX I1 ON TABLE1 (
cont> COLA ASC,
cont> COLB ASC,
cont> COLC ASC)
cont>  TYPE IS SORTED RANKED
cont> NODE SIZE 16300
cont>  USAGE QUERY DUPLICATES ARE COMPRESSED
cont> DISABLE COMPRESSION STORE IN AREA1;
SQL> COMMIT;
$ RMU/VERIFY/INDEX=I1 TESTDB
%RMU-I-BTRDUPCAR, Inconsistent duplicate cardinality (C1) of 150551
                  for entry 1 at dbkey 49:201:0.
                  Actual count of duplicates is 150469
%RMU-I-BTRERPATH, parent B-tree node of 49:201:0 is at 49:5:0
%RMU-I-BTRERPATH, parent B-tree node of 49:5:0 is at 49:933:0
%RMU-I-BTRROODBK, root dbkey of B-tree is 49:933:0
%RMU-I-NDXERRORS,        1 index error encountered
   .
   .
   .
```

Although the index built without a bugcheck error, there was a problem. This occurred when the last entry of a node was a duplicate, the number of duplicates was more than 65536, and the last duplicate was causing the node to split.

As a possbible workaround for this problem, try a different node size or use non-ranked B-tree indexes.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.3.

4.1.6 System Table Change for International Database Users

Prior to this release, an error in the creation of system metadata for storage area map information during database creation caused an incorrect character set to be associated with the RDB$AREA_NAME and RDBVMS$AREA_NAME fields within the RDB$STORAGE_MAP_AREAS and RDBVMS$STORAGE_MAP_AREAS tables respectively.

This problem was only be seen in databases that had a database delimiter character set other than DEC_MCS and usually manifested itself as an error as shown in the following example after trying to access the RDB$AREA_NAME within RDB$STORAGE_MAP_AREAS or RDBVMS$AREA_NAME within RDBVMS$STORAGE_MAP_AREAS tables.

```
%SQL-F-INCCSCMP, Incompatible character set comparison between ...
```

This problem also prevented the new Rdb GUI's, specifically the Schema Manager from viewing indexes and storage maps from Rdb7 databases.

The problem can be corrected by issuing the following SQL script statements after attaching to the affected database.

```
--
--  This SQL script will fix up the area_name fields in both
--  rdb$storage_map_areas and rdbvms$storage_map_areas tables
--  to have the UNSPECIFIED character set ( 32767 )
--

update rdb$field_versions set rdb$field_sub_type = 32767 where
 rdb$field_name = 'RDB$AREA_NAME' and rdb$relation_id =
 ( select rdb$relation_id from
    rdb$relations where rdb$relation_name = 'RDB$STORAGE_MAP_AREAS');

update rdb$field_versions set rdb$field_sub_type = 32767 where
 rdb$field_name = 'RDBVMS$AREA_NAME' and rdb$relation_id =
 ( select rdb$relation_id from
  rdb$relations where rdb$relation_name = 'RDBVMS$STORAGE_MAP_AREAS');
```

This problem has been fixed in Oracle Rdb7 Release 7.0.1.3.

4.1.7 Query Using Outer Zig-Zag Match Strategy with Inner Temporary Table Produced a Bugcheck Error

Bug 626698.

Queries which included temporary tables and caused the Oracle Rdb7 optimizer to chose an outer zig-zag match strategy could cause a bugcheck error if the match keys were of different datatypes.

The following query could produce a bugcheck error:

```
SQL> SELECT
cont> OTHER_NAME,
cont> RDB$CREATED
cont> FROM T1, RDB$RELATIONS
cont> WHERE RDB$RELATION_NAME = NAME ;
!Conjunct
!Match
!  Outer loop      (zig-zag)
!    Get     Retrieval by index of relation RDB$RELATIONS
!      Index name  RDB$REL_REL_NAME_NDX [0:0]
!  Inner loop
!    Temporary relation      Sort    Get
!      Retrieval sequentially of relation ORA_OBJECTS
```

This problem was introduced in Oracle Rdb7 Release 7.0.1.

As a workaround, disable the outer zig-zag match strategy using the following command:

```
$DEFINE RDMS$DISABLE_ZIGZAG_MATCH 1
```

This problem has been corrected in Oracle Rdb7 Release 7.0.1.3.

4.1.8 Queries Where Transitivity Selection Was Not Disabled for Join Predicates Could Cause Bugcheck Errors

Bug 630419.

Queries which contained aggregate subqueries could result in bugcheck errors because the Oracle Rdb7 optimizer did not disable transitivity selection for join predicates.

The following query could cause a bugcheck error:

```
SQL> SELECT CATEGORY_CODE FROM TABLE1
cont> WHERE TABLE_NAME IN
cont>    (SELECT B.TABLE_NAME FROM TABLE1 A, TABLE2 B
cont>     WHERE
cont>     A.FIELD_NAME = '4.0' AND
cont>       A.DOMAIN_NAME = B.TABLE_NAME  AND
cont>       NOT EXISTS
cont>          (SELECT B.FIELD_NAME FROM TABLE1 C
cont>           WHERE
cont>             A.DOMAIN_NAME = C.DOMAIN_NAME ));
```

This problem was introduced in Oracle Rdb7 Release 7.0.1.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.3.

4.1.9 Duplicate Dbkeys Inserted In Wrong Order In Sorted Ranked Indexes

Bug 618553.

This problem pertains to sorted ranked indexes only.

Depending on the history of insertions and removals of DbkeyS within a duplicate entry in a sorted ranked index, it was possible that Dbkeys could be placed in the duplicate entry out of ascending sequence.

While this did not effect the retrieval of these duplicate entries from the index, it was possible that an exception could occur on the modification or deletion of any records that had incorrectly placed Dbkeys within the duplicate entry.

The RMU/VERIFY command did not currently highlight this error because all Dbkeys were present in the index and had corresponding records within the indexed table.

The following scenario describes how this problem occurred:

- A number of duplicate entries were entered for the same entry causing a duplicate overflow node to be created.
- All of the duplicate entries within the primary segment for the entry (the bitmap segment that is held within the entry inside the original index leaf node) were subsequently removed leaving one or more overflow nodes still containing duplicate Dbkeys for this entry.
- A subsequent insertion of a duplicate with a Dbkey with a value that was greater than the reference Dbkey of the first segment within the first overflow node for that entry caused an incorrect insertion of this Dbkey into the primary segment.

A dump listing of an index with this problem showed the following characteristics:

- The Dbkeys with a single duplicate entry were out of ascending order.

- The reference pointer of bitmap segment within the primary entry for the duplicate was the same as or of greater value than the first bitmap segment in the first overflow node for that duplicate.

The following is an extract from the output of a RMU/DUMP/LAREA command for an index that had incorrectly stored duplicate Dbkeys:

```
                                ....  total B-tree node size: 430
                     004F 200D  0092  line 1 (25:27477:1) index: set 79
               0000 FFFFFFFF FFFF  0096  owner 0:-1:-1
                           002C  009E  44 bytes of entries
                           8200  00A0  level 1, full suffix
                  40 00 11 0027  00A2  17 bytes stored, 0 byte prefix
7F00008000028000028000 7F2E008000  00A7   key '................'
                             FF  00B7   key '.'
                    04F6B571 6F  00B8   overflow pointer 79:27478:0
                           005F  00BD   entry cardinality 95.
                           0000  00BF   leaf cardinality 0.
                0085100011 A1  00C1   reference pointer 133:65536:0
                           0005  00C7   5 byte bitmap containing 1 records
               0085 0001087D 0001  00C9    duplicate record 133:67709:1
...

In the overflow node: 79:27478:0

                                ....  total B-tree node size: 430
                     004F 200E  0240  line 0 (25:27478:0) index: set 79
               0000 FFFFFFFF FFFF  0244  owner 0:-1:-1
                           008A  024C  138 bytes of entries
                    04F6B572 6F  0255   overflow pointer 79:27478:1
                0085100011 A1  025E   reference pointer 133:65536:0
                           0074  0264   116 byte bitmap containing 32 records
               0085 000101B2 0001  0266    duplicate record 133:65970:1
               0085 000101B8 0001  026B    duplicate record 133:65976:1
...
```

The only workaround for this problem is to drop and recreate the index as a non-ranked index.

Note

Dropping and recreating the index as a ranked index will remove this problem from the index at that time, however, depending on insertion and removals of duplicates, the problem may re-occur.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.3.

4.1.10 Query Using a Varying Character Datatype as a Join Key Resulted In a Bugcheck Error

Bug 632149.

When the Oracle Rdb7 query optimizer chose a zig-zag match strategy and one of the match keys was defined as a varying character datatype, a bugcheck error could occur.

The following example shows the data definitions and query that could produce this problem:

```
SQL> -- Create tables with character and varying character datatyes
SQL> --
SQL> CREATE TABLE TA (CA CHAR(5));
SQL> CREATE TABLE TB (CB VARCHAR(5));
SQL> INSERT INTO TA VALUES ('A');
SQL> INSERT INTO TA VALUES ('B');
SQL> INSERT INTO TA VALUES ('C');
SQL> INSERT INTO TA VALUES ('C');
SQL> INSERT INTO TA VALUES ('D');
SQL> INSERT INTO TA VALUES ('D');
SQL> INSERT INTO TB VALUES ('B');
SQL> INSERT INTO TB VALUES ('C');
SQL> CREATE INDEX IA ON TA (CA);
SQL> CREATE INDEX IB ON TB (CB);
SQL> COMMIT WORK;
SQL> -- The following select statement will cause a bugcheck error
SQL> SELECT * FROM TA,TB WHERE CA=CB;
```

As a workaround, change the datatype of VARCHAR to CHAR.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.3.

4.1.11 Various Timer Related Problems

Bugs 634135 and 618280.

The following timer related problems were observed in Oracle Rdb7 Release 7.0:

- Processes would sometimes stall in a MUTEX wait state due to exhaustion of TQELM.
- Non-fatal OpenVMS bugchecks would be logged in the system error log due to an EXEC mode timer AST being delivered with an AST address that is no longer valid.
- An ASTFLT or other errors could occur when the following events occurred:
  - An EXEC mode timer AST was queued
  - The process would rundown
  - Another image would be invoked
  - The timer AST would be delivered using an address that might no longer be valid.

Some workarounds to the above problems are:

- Increase TQELM quota to prevent the process from running out of TQELM.
- Ensure there are no severe I/O bottlenecks to the database root file.

These problems have been corrected in Oracle Rdb7 Release 7.0.1.3.

4.1.12 Process Stalls when Starting a NOWAIT Transaction

Bug 635301.

If the FAST COMMIT feature was enabled, it was possible for processes attempting to start a NOWAIT transaction to stall while attempting to obtain the NOWAIT lock. The stall was due to a deadlock between a process that was holding the NOWAIT lock and requesting a lock on a page, and another process that was holding the desired page lock and requesting the NOWAIT lock. The deadlock would not be resolved because Oracle Rdb7 disables deadlock detection on the NOWAIT lock. This was done to prevent the NOWAIT lock from interfering with deadlock resolution for page locks.

To workaround this problem the FAST COMMIT feature may be disabled, or the use of NOWAIT transactions can be avoided.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.3.

4.1.13 Bugcheck Error with Asynchronous Batch Write

A bugcheck error could occur at DIOFETCH$FETCH_SNAP_SEG if a database had asynchronous batch write (ABW) on and its maximum number of pages in a buffer was larger than one and the pages were heavily updated.

The problem was caused by a lost I/O for snapshot updates.

A workaround is to turn off ABW or to change the buffer size in a way that each buffer holds only one page.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.3.

4.1.14 Zig-Zag Match Join Query with Leading NULL Segment Returned Wrong Results

Bug 621750.

The following zig-zag match join query with leading NULL segment returned the wrong results:

```
SQL> SELECT CCN.CONT_ID_NO, CCN.CCN CCN
cont>  FROM CCN, CCN_EXCP CE
cont>  WHERE CCN.CONT_ID_NO IS NULL AND
cont>  CE.CCN = CCN.CCN;
Conjunct
Match
  Outer loop      (zig-zag)
    Index only retrieval of relation CCN
      Index name  CCN00_U2 [1:1]
  Inner loop
    Temporary relation      Sort
    Get     Retrieval sequentially of relation CCN_EXCP
```

A workaround is to define the logical name RDMS$DISABLE_ZIGZAG_MATCH as true.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.3.

4.1.15 Queries Using ORDER BY and GROUP BY Returned Wrong Results

Bug 592375.

The following query using GROUP BY and ORDER BY clauses grouped the results in the wrong order:

```
SQL> SELECT * FROM TAB1 T1, TAB2 T2
cont>  WHERE  T1.PROMO    = T2.PROMO
cont>  GROUP BY T1.ASOC, T1.PROMO
cont>  ORDER BY T1.ASOC, T1.PROMO;
Reduce  Sort
Conjunct
Match
  Outer loop
    Sort    Get     Retrieval sequentially of relation TAB1
  Inner loop
    Temporary relation   Sort  Get     Retrieval sequentially of relation TAB2
 T1.ASOC   T1.PROMO
    600   C027047
    800   C027047
    900   C027047
    800   C055057
    900   C055057
    800   C077067
    900   C077067
   .
   .
   .
```

As a workaround, define an index on TAB1 with the target columns, and an index on TAB2 with the join column.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.3.

4.1.16 Compressed Sorted Index Entry Stored In Incorrect Storage Area

Bug 531995.

Under certain conditions in previous versions of Oracle Rdb, when a partitioned, compressed sorted index was created after data was inserted into a table, B-tree entries could be inserted into the wrong storage area.

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

- The CREATE INDEX command was issued after there were records already in the table on which the index was created
- The index must have been partitioned over a single column
- The index must have had compression enabled
- The scale factor must have been zero on the columns of the index
- No collating sequences were specified on the columns of the index
- No descending indexes existed
- Mapping values must not have been specified

The RMU/DUMP/AREA=xx command would show that the B-tree entry was not stored in the expected storage area. However, in previous versions of Oracle Rdb, the rows of the table could still be successfully retrieved.

The following example shows the problem:

```
SQL> CREATE DATABASE FILENAME FOO
cont> CREATE STORAGE AREA AREA_1
cont> FILENAME AREA_1
cont> CREATE STORAGE AREA AREA_2
cont> FILENAME AREA2;
SQL>
SQL> CREATE TABLE t1
cont> (C1 INTEGER);
SQL>
SQL> -- Insert data into table prior to index creation
SQL> INSERT INTO T1 VALUES (0);
SQL> COMMIT;
SQL>
SQL> -- Create index with COMPRESSION ENABLED
cont> CREATE INDEX INDEX_1
cont>    ON T1 (C1)
cont>    ENABLE COMPRESSION
cont>    STORE USING (C1)
cont> IN AREA_1 WITH LIMIT OF (0)
cont> OTHERWISE IN AREA_2;
SQL> COMMIT;

--  Dump out the page for the B-tree in AREA_1. There are 0 bytes stored.
--  There should be 5 bytes stored for this B-tree entry.

RMU/DUMP/AREA=AREA_1
.
.
.                                ....  total B-tree node size: 430
                       0030 2003  0240  line 0 (2:5:0) index: set 48
              002F FFFFFFFF FFFF  0244  owner 47:-1:-1
                            0000  024C  0 bytes of entries <---***** no entry
                            8200  024E  level 1, full suffix
00000000000000000000000000000000  0250  unused '................'
.
.
.

 Dump out the page for b-tree in AREA_2, there are 5 bytes stored

RMU/DUMP/AREA=AREA_2
.
.
.                                ....  total B-tree node size: 430
                       0031 2003  0240  line 0 (3:5:0) index: set 49
              002F FFFFFFFF FFFF  0244  owner 47:-1:-1
                            000A  024C  10 bytes of entries
                            8200  024E  level 1, full suffix
                           00 05  0250  5 bytes stored, 0 byte prefix <---entry
                     0100008000  0252   key '.....'
                        22B1 10  0257   pointer 47:554:0
.
.
.
```

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

In prior versions, the rows were successfully retrieved even though the key values were stored in the wrong storage area. This was due to the range query algorithm skipping empty partitions or scanning extra areas. However, due to an enhancement in the algorithm for range queries on partitioned sorted indexes in Oracle Rdb Release 7.0.1.3, the rows of the table which are stored in the incorrect storage areas may not be retrieved when using the partitioned index.

The optimized algorithm now only scans the relevant index areas and no longer skips over empty areas, resulting in only those rows being returned.

Therefore, it is recommended that the index be dropped and re-created or alternatively, as a short term solution, the new optimization be disabled by defining the logical name RDMS$USE_OLD_INDEX_PART_CHECK to 1.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.3.

4.1.17 ALTER STORAGE MAP ... DISABLE COMPRESSION Corrupted Some Areas

In prior versions of Oracle Rdb7, an ALTER STORAGE MAP ... DISABLE COMPRESSION statement incorrectly processed partitions in a storage map with more than one storage area. Because of this, some of the partitions were processed twice which also attempted to disable compression twice. This situation caused incorrect results to be returned from queries on these partitions and, in some cases, the ALTER STORAGE MAP statement returned a bugcheck error.

A workaround is to drop and recreate the table when changing the compression characteristic.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.3.

4.1.18 DROP STORAGE AREA CASCADE Involving a Ranked Index Could Cause a Bugcheck Error

Bug 616535.

In previous versions of Oracle Rdb7, dropping a storage area that included ranked indexes using the CASCADE qualifier could result in a bugcheck error.

The following example demonstrates this problem:

```
SQL> CREATE DATABASE FILE TESTDB70.RDB
cont> CREATE STORAGE AREA RDB$SYSTEM          FILE 'TESTDB70_RDB_SYSTEM'
cont> CREATE STORAGE AREA TESTDB70_DATA_1     FILE 'TESTDB70_DATA_1'
cont> CREATE STORAGE AREA TESTDB70_INDEXES_1  FILE 'TESTDB70_INDEXES_1'
cont> CREATE STORAGE AREA TESTDB70_DATA_2     FILE 'TESTDB70_DATA_2'
cont> CREATE STORAGE AREA TESTDB70_INDEXES_2  FILE 'TESTDB70_INDEXES_2';
SQL>
SQL> CREATE TABLE TAB1 ( COL1 INTEGER, COL2 CHAR(10), COL3 INTEGER );
SQL>
SQL> CREATE STORAGE MAP TAB1_MAP FOR TAB1
cont> PARTITIONING IS NOT UPDATABLE
cont>  STORE USING ( COL1 )
cont>  IN TESTDB70_DATA_1 WITH LIMIT OF ( 1000 )
cont>  IN TESTDB70_DATA_2 WITH LIMIT OF ( 2000 );
SQL> -- Create a sorted ranked index
SQL>
SQL> CREATE UNIQUE INDEX TAB1_SNDX ON TAB1(COL1)
cont> TYPE IS SORTED RANKED
cont> STORE USING ( COL1 )
cont> IN TESTDB70_INDEXES_1 WITH LIMIT OF ( 1000 )
cont> IN TESTDB70_INDEXES_2 WITH LIMIT OF ( 2000 );
SQL>
SQL> -- Insert data to table tab1 and commit.
SQL>
SQL> -- Drop the storage area containing the index
SQL>
SQL> ALTER DATABASE FILE TESTDB70
cont>  DROP STORAGE AREA TESTDB70_DATA_1
cont>  CASCADE;
```

The following exception error is generated:

```
PSIISCAN$GET_NEXT_UNIQUE + XXX
```

This problem was caused because non-ranked sorted index scan code was invoked rather than the ranked index scan code.

As a possible workaround for this problem, use a non-ranked index.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.3.

4.1.19 Creating a Ranked Index After Tables Were Loaded Could Cause a Bugcheck Error

Bugs 642381 and 638598.

In Oracle Rdb7 Release 7.0.1.2, a problem was introduced where creating a ranked index after tables had been loaded could cause a bugcheck error.

The following example demonstrates this behavior:

```
SQL> -- After the table CMFDATCPT is populated with lots of data.
SQL>
SQL> CREATE INDEX CMFDATCPT_I5
cont>    ON CMFDATCPT ( TYP_ENREG asc,
cont>    GEN_TRAIT asc, DAT_COMPT asc)
cont>    TYPE IS SORTED RANKED
cont>    NODE SIZE 960
cont>    PERCENT FILL 60
cont>    DISABLE COMPRESSION STORE IN DATCPT_1_IDX ;
```

This problem could produce one of the following error messages:

```
PSIIBUILD2BLDBBCDUP + xxx
```

or

```
PSIIBUILD2BUILDFROMBOTTOM + xxx
```

As a possible workaround to this problem, use non-ranked indexes.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.3.

4.1.20 A SELECT Query Involving a Compressed and Uncompressed Index Could Produce a Bugcheck Error

Bugs 647888 and 636860.

When a compressed sorted ranked index and another sorted ranked index were used by the dynamic optimizer in a query, the stack could become corrupted and a bugcheck error could result. This was caused by the optimizer's incorrect assumption that all index keys processed in a particular phase are compressed. The attempt to uncompress an already uncompressed key caused the stack corruption. The database was not corrupted by this action.

The following example demonstrates this problem:

```
-- Populate table ASSET_ACCOUNT with data.
SQL>  CREATE INDEX ACC_TY_CO_IND ON ASSET_ACCOUNT (ACCOUNT_TYPE_CODE)
cont>    TYPE IS SORTED RANKED ENABLE COMPRESSION;
SQL>
SQL>  CREATE INDEX ID_IND ON ASSET_ACCOUNT (ID) TYPE IS SORTED;
SQL>
SQL>  SELECT NAME INTO :NAME FROM ASSET_ACCOUNT
cont>                        WHERE ID                 = :FIRM
cont>                          AND ACCOUNT_TYPE_CODE  = 'PB'
cont>                          AND FUNDS_SEG_TYP_CODE = :SEG_CODE
cont>                          AND BUS_FUNC_CODE      = 'CLR';
```

The following bugcheck error could result:

```
Exception at 20202020 : symbol not found
Saved PC = 00A33E1C : PSII2ESTIMATECARD + xxx
```

As a possible workaround for this problem, use a non-ranked sorted index or use a sorted ranked index with compression disabled.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.3.

4.1.21 Incorrect Results from SORTED RANKED Index During "Direct Key Lookup"

Bug 636630.

In prior releases of Oracle Rdb7, it was possible for a query to return incorrect results if the following were true:

- The dynamic optimizer was used (Leaf strategy)
- One of the selected indexes was a UNIQUE SORTED RANKED index
- The query used a "direct key lookup" access method
- The dynamic optimizer chose a "ZeroShortCut" method

The estimation phase of the dynamic optimization (see the output of the RDMS$DEBUG_FLAGS "E", or the SET FLAGS 'EXECUTION') incorrectly returned a zero estimate when there was a direct key lookup. This sometimes caused the "ZeroShortCut" method to be selected and no disk I/O was performed.

This problem has been corrected and the estimate returned for sorted ranked indexes has been improved.

Note

For this release the estimation output is changed slightly to report statistics for sorted ranked indexes which, in prior releases, were always zero. The Ndx:Lev/Seps/DBKeys output reports the index number (Ndx), the average number of leaf nodes (Lev), the minimum number of Dbkeys (Seps), and the average number of Dbkeys (DBKeys).

As a workaround, redefine the sorted ranked index as non-unique. Alternatively, a unique sorted index could be used to replace the sorted ranked index.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.3.

4.1.22 RDMAIJSERVER Account Priority Set to Fifteen

All OpenVMS platforms.

When the Hot Standby feature is installed, the RDMAIJSERVER account is created by the installation procedure. The installation procedure did not set the account's priority correctly resulting in the UAF default value (generally 4) being used.

Running at this priority could, on a busy system, cause the network link to become "full" because the AIJ server process was not reading the network fast enough. This could lead to slowdowns on the master database.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.3. The RDMAIJSERVER account is now created specifying an account priority of 15. Any existing RDMAIJSERVER account will be modified to specify the priority of 15.

The priority of AIJ server processes on your system can be restricted with the system-wide logical name RDM$BIND_AIJSRV_PRIORITY. If this logical name is defined to a value less than 15, an AIJ server process will adjust its base priority to the value specified when the AIJ server process starts. Values from 0 to 31 are allowed for RDM$BIND_AIJSRV_PRIORITY, but the process is not able to raise its priority above the RDMAIJSERVER account value.

4.1.23 Query Returned Wrong Result when German Collating Sequence Defined

Bug 653283.

The following query with match strategy returned the wrong results when the German collating sequence was defined.

```
SQL> SELECT TABLE_B.CODE
cont> FROM TABLE_A, TABLE_B, TABLE_C, TABLE_D
cont> WHERE
cont>    TABLE_A.NUMBER = TABLE_B.NUMBER
cont>    AND TABLE_A.NUMBER = TABLE_C.NUMBER
cont>    AND TABLE_B.CODE = TABLE_D.CODE ;
Conjunct
Match
  Outer loop
    Sort    Conjunct
    Match
      Outer loop
        Conjunct
        Match
          Outer loop      (zig-zag)
            Get     Retrieval by index of relation TABLE_A
              Index name  INDEX_1 [0:0]
          Inner loop      (zig-zag)
            Index only retrieval of relation TABLE_C
              Index name  INDEX_2 [0:0]
      Inner loop      (zig-zag)
        Get     Retrieval by index of relation TABLE_B
          Index name  INDEX_3 [0:0]
  Inner loop      (zig-zag)
    Get     Retrieval by index of relation TABLE_D
      Index name  INDEX_4 [0:0]
 TABLE_B.CODE   TABLE_A.NUMBER
 aaaaa                      1
1 row selected.
```

The only workaround to this problem is to disable the collating sequence.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.3.

4.1.24 Index Prefix Cardinalities Not Set to Zero After TRUNCATE TABLE

Bug 644764.

In prior releases of Oracle Rdb7, a TRUNCATE TABLE command, once committed, did not correctly reset the index prefix cardinalities to zero.

The workaround is to issue a DELETE statement instead of TRUNCATE TABLE.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.3.

4.1.25 Bugcheck Error at RDMS$$CREATE_ETRG + 0000144C

Bug 652110.

OpenVMS Alpha platforms.

In rare cases, Oracle Rdb would generate a bugcheck error with the following exception when a trigger action was executed.

```
**** Exception at 00DBB814 : RDMS$$CREATE_ETRG + 0000144C
%SYSTEM_F_ACCVIO
```

Virtual memory used by the trigger request was prematurely freed. The memory was then subsequently referenced which resulted in an access violation and bugcheck dump.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.3.

4.1.26 Query Using a Table with Many Indexes Ran Out of Memory Quota

Bug 658882.

When a table had many indexes defined on it, a simple SELECT query with an equality selection ran out of memory with the following error:

```
%COSI-F-EXQUOTA, exceeded quota
-SYSTEM-F-EXQUOTA, process quota exceeded
```

The following example shows this problem where the optimizer repeatedly created redundant ordering data structures for the order "FIRST_NAME, LAST_NAME" and finally ran out of memory if the number of indexes increases.

```
SQL> CREATE INDEX EMP_F_L ON EMPLOYEES (FIRST_NAME, LAST_NAME);
SQL> CREATE INDEX EMP_F_M ON EMPLOYEES (FIRST_NAME, MIDDLE_INITIAL);
SQL> CREATE INDEX EMP_F_C ON EMPLOYEES (FIRST_NAME, CITY);
SQL> CREATE INDEX EMP_F_S ON EMPLOYEES (FIRST_NAME, STATE);
SQL> CREATE INDEX EMP_F_C_S ON EMPLOYEES (FIRST_NAME, CITY, STATE);
SQL>
SQL> SELECT EMPLOYEE_ID FROM EMPLOYEES
cont>    WHERE FIRST_NAME = 'Foo' AND LAST_NAME = 'Bar'
cont>    ORDER BY FIRST_NAME, LAST_NAME;
%COSI-F-EXQUOTA, exceeded quota
-SYSTEM-F-EXQUOTA, process quota exceeded
```

As a workaround drop enough indexes to run the query.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.3.

4.1.27 Read-only Transactions Fetched AIP Pages Too Often

Oracle Rdb7 read-only transactions fetch area inventory pages (AIP) to ensure that the logical area has not been modified by an exclusive read/write transaction. This check is needed because an exclusive read/write transaction does not write snapshot pages and these pages may be needed by the read-only transaction.

Because AIPs are always stored in the RDB$SYSTEM area, reading the AIP pages could represent a significant amount of I/O to the RDB$SYSTEM area for some applications. Setting the RDB$SYSTEM area to read-only can avoid this problem, but it also prevents other online operations that might be required by the application so it is not a viable workaround in all cases.

This problem has been reduced in Oracle Rdb7 Release 7.0.1.3. The AIP entries are now read once and are not read again unless they need to be. This optimization requires that the carry-over locks feature be enabled (the default setting). If carry-over locks are not enabled, this optimization is not enabled and the behavior is the same as in prior versions.

4.1.28 Not All Rows Returned from Sequential Scan

Bug 668246.

Oracle Rdb did not return all rows from a table when more than one area bit map (ABM) page was needed to represent all of the space area management (SPAM) pages for the table. There was an error in the scan algorithm that prevented a sequential scan from seeing any ABM entries in any ABM page beyond the first page for a logical area (table or partition within a table). After all rows had been returned that were represented by the first ABM page, no further rows were returned.

At the same time the RMU/VERIFY command returned erroneous verify errors for the logical area. For example, when using Rdb Release 6.1, RMU/VERIFY would report:

```
%RMU-I-BGNABMSPM, beginning ABM pages verification
%RMU-I-OPENAREA,  opened storage area AREA_1 for protected retrieval
%RMU-W-BADABMIND, max set bit index of area bit map page 2
                  for logical area 48 out of range
                  expected to be in range 0 : 3680, found: 3865
%RMU-W-BADABMIND, max set bit index of area bit map page 4
                  for logical area 50 out of range
                  expected to be in range 0 : 3680, found: 3866
%RMU-E-BADABMPAG,       error verifying ABM pages
%RMU-I-ENDABMSPM, completed ABM pages verification
```

RMU/VERIFY was using 3680 entries as the maximum number of entries allowed when the correct number was actually 7360.

When using Oracle Rdb7 on the same database, RMU/VERIFY would report errors similar to the following:

```
%RMU-I-BGNABMSPM, beginning ABM pages verification
%RMU-W-ABMBITERR, inconsistency between spam page
1431521 and bit 3681 in area bitmap in larea 48 page 2
%RMU-W-ABMBITERR, inconsistency between spam page
1503875 and bit 186 in area bitmap in larea 48 page 3
%RMU-W-ABMBITERR, inconsistency between spam page
1503486 and bit 185 in area bitmap in larea 50 page 5
%RMU-W-ABMBITERR, inconsistency between spam page
1504264 and bit 187 in area bitmap in larea 50 page 5
%RMU-E-BADABMPAG,       error verifying ABM pages
%RMU-I-ENDABMSPM, completed ABM pages verification
```

This problem would only affect tables that were stored in a uniform-format storage area. The problem would only occur when more than one ABM page was needed to represent a logical area. It was most likely to be seen when using a small database page size, for example, one block pages, or when using a small SPAM interval in a storage area.

Prior releases of Oracle Rdb7 could retrieve the data using an index if the index existed prior to the table growing to the point that it utilized more than one ABM page. Any index built after the table required more than one ABM page would not contain entries for all rows in the table.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.3.

This release enables Oracle Rdb to retrieve the data using a sequential scan retrieval strategy.

4.1.29 Extra I/O with Query Using Sorted Duplicate Index

Bug 647454.

In prior releases of Oracle Rdb7, additional I/O occurred when a query involving a literal value used a sorted duplicate index containing many duplicate values.

Prior to Oracle Rdb7, small valued numeric literals (between -32768 and 32767) were accepted as a SMALLINT datatype. With Oracle Rdb7, these small integers are promoted to INTEGER for better performance on VAX and Alpha platforms. This change forced the optimizer to widen the search in case the INTEGER was larger than what would fit into a SMALLINT datatype and therefore resulted in an extra "Conjunct" in the strategy and the extra I/O.

The following example displays the problem:

```
SQL> CREATE TABLE T (C SMALLINT);
SQL> CREATE INDEX I ON T (C);
SQL> -- insert many of the same values into t
SQL> INSERT INTO T(C) VALUES (33);
SQL> INSERT INTO T(C) VALUES (33);
SQL> INSERT INTO T(C) VALUES (33);
.
.
.
SQL> COMMIT;
SQL>
SQL> -- The following query results in a strategy involving a "Conjunct" and
SQL> -- forces the optimizer to search the duplicates chain of values
SQL> -- including value "33"
SQL>
SQL> SELECT COUNT(*) FROM T WHERE C < 33;
Aggregate       Conjunct        Index only retrieval of relation T
  Index name  I [0:1]
```

```
          0
1 row selected
```

The workaround is to use CAST in the query as follows:

```
SQL> SELECT COUNT(*) FROM T WHERE C < CAST (33 as SMALLINT);
```

This problem has been corrected in Oracle Rdb7 Release 7.0.1.3.

4.1.30 Rows Missing In Recovered Database After Verb Failure

When a verb failure occurred while storing a row, the row could be left in a locked state with incorrect TSN values. Another process could then store a record with the same Dbkey value as the locked row. Depending on the sequence of transaction commits between the two processes, it was possible that the rows stored would be missing from the database if it was subsequently restored and recovered.

The following script shows this problem using two processes (P1 and P2) and a single database.

```
P1>  $! Create the database with two tables in a mixed
P1>  $! page format storage area with a unique index on
P1>  $! table T1.  Insert a row into table T1 with a key
P1>  $! value of 1.
P1>  $!
P1>  $ sql$
SQL>  CREATE DATA FILE 'T'
cont>    NUMBER OF BUFFERS 2
cont>    CREATE STORAGE AREA RDB$SYSTEM FILENAME RDBSYS
cont>    CREATE STORAGE AREA S FILENAME S PAGE FORMAT MIXED;
SQL>  CREATE TABLE T1 (I INTEGER);
SQL>  CREATE TABLE T2 (I INTEGER);
SQL>  CREATE UNIQUE INDEX I1 ON T1 (I) TYPE IS SORTED;
SQL>  CREATE STORAGE MAP M1 FOR T1 STORE IN S;
SQL>  CREATE STORAGE MAP M2 FOR T2 STORE IN S;
SQL>  INSERT INTO T1 VALUES (1) RETURNING DBKEY;
SQL>  COMMIT;
SQL>  DISCONNECT ALL;
SQL>  ALTER DATA FILE T ADD JOURNAL J FILE J JOURNAL ENABLE;
SQL>  EXIT;

P1>  $! Backup the database and the AIJ.
P1>  $!
P1>  $ RMU /BACKUP T.RDB T.RBF
P1>  $ RMU /BACKUP /AFTER /QUIET T.RDB T.AIJ-BCK

P1>  $! Attempt to insert a second record into table
P1>  $! T1 again with a key value of 1.  The insert
P1>  $! fails due to a duplicate key value.  Perform
P1>  $! additional operations to cause the database
P1>  $! page to be flushed back to disk.
P1>  $!
P1>  $ SQL$
SQL>  ATTACH 'FILE T';
SQL>  INSERT INTO T1 VALUES (1);
SQL>  SHOW TABLES;
```

```
   .
   .
   .
SQL>  SHOW STORAGE AREAS;
   .
   .
   .
P2>  $! From another process, insert a new row into
P2>  $! table T2 then commit and exit.
P2>  $!
P2>  $ SQL$
SQL>  ATTACH 'FILE T';
SQL>  INSERT INTO T2 VALUES (2) RETURNING DBKEY;
SQL>  COMMIT;

P1>  $! Commit the transaction that performed the
P1>  $! original insert.  Select data from table
P1>  $! T2 to show that it is in the database.
P1>  $!
SQL>  COMMIT;
SQL>  SELECT I, DBKEY FROM T2;
   .
   .
   .
SQL>  EXIT;

P1>  $! Restore and recover the database.
P1>  $!
P1>  $ RMU /RESTORE /NOCDD /NOACL /NORECOVER /NEW_VERSION T.RBF
P1>  $ RMU /RECOVER /ROOT=T.RDB;0 /LOG /TRACE J.AIJ
P1>
P1>  $! Attached to the restored database and select
P1>  $! the data from table T2.  There are no rows in
P1>  $! the table after the recovery.
P1>  $!
P1>  $ SQL$
SQL>  ATTACH 'FILE T';
SQL>  SELECT I, DBKEY FROM T2;
   .
   .
   .
SQL>  EXIT;
```

This problem has been corrected in Oracle Rdb7 Release 7.0.1.3. A verb failure during a store operation now sets the TSN of the row to the correct value preventing reuse of the row.

4.1.31 Index Segment Prefix Cardinality Set to Zero

Bug 644764.

In prior versions of Oracle Rdb7, it was possible for a query strategy to change if the prefix cardinality of any index segment prior to the last was set to zero. If the leading segments of a multisegment index had very low cardinalities it was possible, during delete operations, to decrement the cardinality to zero. This resulted in an estimation of the segment cardinality which lead to an inappropriate index being chosen.

As an example, create a table index like the following:

- The leading segment contains only one value
- The next segment contains 3 values
- The next segment contains 4 values

- The last segment contains hundreds of values

This type of index could generate the following cardinalities:

```
                  Cardinality
segment# 0 1
segment# 1 3
segment# 3 12
segment# 4 0
```

This data would have a common value for the leading 3 segments spread over many of the level 1 nodes. If a delete operation took place, as the keys in a level 1 node were deleted, the node would become empty and then be deleted. When this level 1 node was deleted, a status indicating that this was the last value of the key found would be returned and the prefix cardinality of the segments would be decremented.

As deletions continued the following could result:

```
                  Cardinality
segment# 0 0
segment# 1 0
segment# 3 8
segment# 4 0
```

When the optimizer determined the cost of the prefix cardinalities, other than the last, were zero an estimation of the cardinalities was used. This created estimation errors associated with the data distributions which lead to an inappropriate index being selected for the query.

This behavior is associated with the design of the index.

The only workaround is to reset the cardinalities by running the RMU/COLLECT OPTIMIZER_STATISTICS command. This is an online function, please refer to the help for "RMU Collect_Optimizer_Statistics" for an explanation of the /TRANSACTION_TYPE qualifier.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.3. When a level one node is deleted, the associated segment cardinality is not decremented.

It is recommended that the RMU/COLLECT OPTIMIZER_STATISTICS /INDEXES/NOTABLES command be run after installation of the MUP to ensure that the prefix cardinalities are correct.

4.1.32 Dbkeys Were Not Reused If Snapshots Were Disabled

Bugs 674348 and 669572.

In previous versions of Oracle Rdb7, if snapshots were disabled, Dbkeys were not reused when they should have been. If one user deleted records and then detached from the database, another user should have been able to reuse the Dbkeys of deleted records. This did not happen and resulted in many pages of the database having large number of deleted lines with no locked free space assigned to them.

This could manifest itself in an excessive number of pages being checked when a user was trying to store records.

The following example shows the problem:

```
SQL> CREATE DATABASE FILENAME REPROD SNAPSHOTS DISABLED;
SQL> CREATE TABLE TAB1(F1 INTEGER);
SQL> INSERT INTO TAB1 VALUES (1);
1 row inserted
SQL> INSERT INTO TAB1 VALUES (1);
1 row inserted
SQL> INSERT INTO TAB1 VALUES (1);
1 row inserted
SQL> INSERT INTO TAB1 VALUES (1);
1 row inserted
SQL> INSERT INTO TAB1 VALUES (1);
1 row inserted
SQL> SELECT DBKEY FROM TAB1;
                  DBKEY
               47:554:0
               47:554:1
               47:554:2
               47:554:3
               47:554:4
5 rows selected
SQL> COMMIT;
SQL> DELETE FROM TAB1;
5 rows deleted
SQL> COMMIT;
SQL> DISCONNECT ALL;
SQL> -- Note that if the inserts were done in the same attach then the Dbkeys would
SQL> -- be reused.
SQL> ATTACH 'FILENAME REPROD';
SQL> INSERT INTO TAB1 VALUES (1);
1 row inserted
SQL> INSERT INTO TAB1 VALUES (1);
1 row inserted
SQL> INSERT INTO TAB1 VALUES (1);
1 row inserted
SQL> INSERT INTO TAB1 VALUES (1);
1 row inserted
SQL> INSERT INTO TAB1 VALUES (1);
1 row inserted
SQL> COMMIT;
SQL> -- Dbkeys 0-4 should be reused on this page but are not.
SQL> SELECT DBKEY FROM TAB1;
                  DBKEY
               47:554:5
               47:554:6
               47:554:7
               47:554:8
               47:554:9
5 rows selected
SQL> EXIT
```

The workaround to this problem is to either enable snapshots or enable them as DEFERRED. Then Dbkeys are reused as expected.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.3.

4.1.33 Join of Two Tables Resulted In RDMS$$EXE_NEXT Bugcheck Error

Bugs 671153, 626542 and 707463.

A query that made an outer join on two tables, one of which was empty and one of which contained data, could result in a bugcheck with the exception message:

```
***** Exception at 0060E750 : RDMS$$EXE_NEXT + 00000033
%COSI-F-BUGCHECK, internal consistency failure
```

This offset value is for a VAX running Rdb Release 7.0.1.2. Other releases of Rdb7 on other platforms would have different values.

The following is an example of the type of query that would create the problem:

```
SQL> SELECT C1.F1, C3.F1, C1.F2, C3.F2, C1.F3, C3.F3, C1.F4, C3.F4,
cont> C1.F5, C3.F5
cont> FROM
cont> (SELECT
cont>     DISTINCT C4.CP_TYPE, C4.GL_ACCT, C4.GL_DEPT, C4.CP_LINE_CD, 1
cont>     FROM CP_FM C4) as C3 ( F1, F2, F3, F4, F5 )
cont>        FULL OUTER JOIN
cont>        (SELECT
cont>             DISTINCT C2.CP_TYPE, C2.GL_ACCT, C2.GL_DEPT, C2.CP_LINE_CD, 1
cont>             FROM CP_HIST C2) AS C1 ( F1, F2, F3, F4, F5 )
cont>           ON ((((C1.F1 = C3.F1)
cont>                  AND (C1.F2 = C3.F2))
cont>                  AND (C1.F3 = C3.F3))
cont>                  AND (C1.F4 = C3.F4));
```

The only workaround is to ensure that both tables contain data.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.3.

4.2 SQL Errors Fixed

4.2.1 Storage Map Compression Option Not Exported and Imported Correctly

In prior releases of Oracle Rdb7, it was possible that the compression option for a storage map was not exported and imported correctly. For example, a storage map with compression disabled would, after being exported and imported have compression enabled. This is shown in the following example:

```
SQL> CREATE TABLE T (ID INT);
SQL> CREATE STORAGE MAP M FOR T DISABLE COMPRESSION;
SQL> COMMIT;
SQL> EXPORT DATA FILE STO_MAP INTO STO_MAP;
SQL> IMPORT DATA FROM STO_MAP FILE STO_MAP;
Exported by Oracle Rdb X7.0-00 Import/Export utility
A component of Oracle Rdb SQL X7.0-00
Previous name was sto_map
It was logically exported on  6-JUL-1998 13:15
Multischema mode is DISABLED
Database NUMBER OF USERS is 50
Database NUMBER OF CLUSTER NODES is 16
Database NUMBER OF DBR BUFFERS is 20
Database SNAPSHOT is ENABLED
Database SNAPSHOT is IMMEDIATE
   .
   .
   .
 Compression is: ENABLED
```

A suggested workaround is to use the RMU/EXTRACT/ITEM=STORAGE_MAP command and edit the output into your IMPORT command, so that each storage map is fully specified by IMPORT.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.3.

4.2.2 CREATE STORAGE MAP with COLUMNS clause may delete process

Bug 693696

In prior releases of Oracle Rdb7 it was possible that the CREATE STORAGE MAP command for a vertically partitioned table (using the COLUMNS clause) would generate an error such as that shown below:

```
%COSI-F-EXQUOTA, exceeded quota
-SYSTEM-F-VASFULL, virtual address space is full
```

In rare cases it could cause the current process to be deleted with accounting reporting an SS$_ACCVIO process termination status.

The problem occurred while processing very long lists of columns in the COLUMNS clause, where the total length of the column names plus overhead exceeded 1024 bytes. The only workaround for this problem is to limit the total length of the columns names to less than 1024 bytes.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.3.

4.2.3 Unexpected Errors and Bugchecks when Calling Stored Procedures

Bug 634943.

In previous versions of Oracle Rdb7 (Release 7.0.1.1 and Release 7.0.1.2) a problem was introduced which could cause a bugcheck or an unexpected error when running a stored procedure. Two of the reported errors are shown below.

Example 4–1 Bugcheck Error In RDMS$$PRE_EXECUTION (Alpha OpenVMS)

```
***** Exception at 00FF2C20 : RDMS$$PRE_EXECUTION + 000002E0
%SYSTEM-F-ACCVIO, access violation, reason mask=00, virtual address=000000000
000002C, PC=0000000000FF2C20, PS=0000000B
```

Example 4–2 Unexpected ARITH_EXCEPT Error (VAX OpenVMS)

```
%RDB-E-ARITH_EXCEPT, truncation of a numeric value at runtime
-SYSTEM-F-ROPRAND, reserved operand fault at PC=0068EA3A, PSL=01C00000
```

For this problem to occur, the stored procedure or stored function must have had more than 64 parameters and/or variables in the routine and these variables must have included OUT or INOUT parameters or updatable variables. For instance, a routine with 65 parameters which updated the last parameter could encounter this problem. Another example would be a stored routine with 10 parameters and more than 54 variables and one of the last declared variables was updated.

The problem could also occur within a multistatement procedure (an anonymous non-stored routine) which had more than 64 variables declared. If the count of the variables and parameters of the routine was less than or equal to 64 then this problem did not occur.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.3.

4.2.4 CREATE INDEX Converted SORTED to SORTED RANKED when DUPLICATES Clause Used

Bug 574085.

In prior releases of Oracle Rdb7, the type of sorted index was changed from the specified SORTED type to SORTED RANKED if you specified the DUPLICATES ARE COMPRESSED clause. No warning was given for this change and this action was confusing to database administrators.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.3.

Users will not be permitted to use the DUPLICATES ARE COMPRESSED clause for TYPE IS SORTED indexes. This clause will only be used when the index type is SORTED RANKED.

The following example shows the error message that will be displayed:

```
SQL> CREATE INDEX IDX1 ON EMPLOYEES(LAST_NAME)
cont>    TYPE IS SORTED
cont>    DUPLICATES ARE COMPRESSED;
%SQL-F-CONFLATTR, conflicting attributes specified: TYPE SORTED and DUPLICATES
```

4.2.5 Interactive SQL No Longer Prompts After ALTER INDEX ... MAINTENANCE IS DISABLED

Bug 606825.

In prior releases of Oracle Rdb, interactive SQL issued a confirmation prompt when the ALTER INDEX command was used to disable maintenance as in the following example.

```
SQL> ALTER INDEX EMPLOYEES_HASH MAINTENANCE IS DISABLED;
This index was previously specified with a STORE clause. Continue? [N]
%SQL-F-CHGINDMAPSTP, Terminating operation at user's request
```

This warning was intended to warn database administrators who inadvertently altered an index without also specifying the STORE clause. Doing so caused the index to be remapped to the default storage area, which was not what was intended.

However, when the clause MAINTENANCE IS DISABLED is used, this warning is not needed because no remapping is performed and it may have been confusing. This problem has been corrected in Oracle Rdb7 Release 7.0.1.3.

4.2.6 INTOVF Error Reported for Some Queries

Bugs 566129 and 562247.

Prior versions of Rdb7 could return an exception when a UNION, a UNION ALL, or a variation of the CASE statement (which includes COALESCE, NULLIF, NVL or DECODE) were used in an expression. This error message is shown in the following example:

```
%RDB-E-ARITH_EXCEPT, truncation of a numeric value at runtime
-COSI-F-INTOVF, integer overflow
```

The problem occurred for UNION or UNION ALL when one column expression was a BIGINT type and the corresponding column expression was a small integer literal value. This caused Rdb to assigned a type of INTEGER for the select column expression, even when this type could not hold the resulting value. The small integer literal needed to be in the range -32768 and 32767. These are the values that can be stored in a SMALLINT datatype.

This problem could occur in one of the various CASE expression variants when one branch resulted in a small integer literal (with the same range as described above) and another branch resulted in a BIGINT result. Rdb would then incorrectly assign a small precision to the result. When the case expression was then included in another expression, such as a subtraction, the resulting datatype assigned to the expression was too small for the result of the expression evaluation.

In both cases, replacing the small integer literals with a CAST expression can avoid this problem. For instance, if the small integer literal were zero (0), then this could be replaced by CAST(0 as INTEGER).

This problem has been corrected in Oracle Rdb7 Release 7.0.1.3. This problem does not occur if small valued integer literals do not appear in the UNION column select list, or as case expression results.

4.2.7 Looping Error Message SQL$_CMPBYWNRL

Bug 572514.

Under certain conditions when processing a CREATE TRANSFER statement, SQL would repeatedly issue the following error message:

```
"SQL$_CMPBYWNRL, Invalid computed field
<column-name> will not be transferred from
relation <table-name>"
```

This message indicated that the named table column could be transferred by the Replication Option for Rdb. In this case, SQL would remove the offending column from the transfer definition after issuing the warning message. When two or more such problem columns happened to be the last in the list of columns for a given table, SQL would repeat the warning indefinitely.

The following example shows a table definition and CREATE TRANSFER statement which would have resulted in the endless warning messages from SQL.

```
SQL> ATTACH 'FILE DISK:[DIR]SOURCE.RDB';
SQL>
SQL> -- Create a table with one good column followed by two computed columns that
SQL> -- are not acceptable for transfer.
SQL>
SQL> CREATE TABLE TAB1 (
cont> COL1 INTEGER,
cont>     COL2  COMPUTED BY
cont>  ( SELECT MAX (COL1) FROM TAB1 ),
cont> COL3 COMPUTED BY
cont>  ( SELECT AVG (COL1) FROM TAB1 ) );
cont> COMMIT;
SQL>
SQL> -- The following transfer definition repeatedly displayed a SQL warning
SQL> -- message:  %SQL-W-CMPBYWNRL, Invalid computed field COL3 will not be
SQL> -- transferred from relation TAB1.
SQL>
SQL> CREATE TRANSFER ENDLESS_WARNING TYPE IS EXTRACTION
cont>      MOVE TABLES TAB1
cont>      TO EXISTING FILENAME DISK:[DIR]TARGET.RDB
cont>      LOGFILE IS DISK:[DIR]ENDLESS_WARNING.LOG;
```

This problem has been corrected in Oracle Rdb7 Release 7.0.1.3. That is, when an invalid column is detected, SQL will output a warning message, but only one message per column.

4.2.8 Bugcheck During DROP MODULE Statement

It was possible in previous versions of Oracle Rdb7 to produce a bugcheck error while executing the DROP MODULE statement. This problem could occur on either VAX or Alpha platforms giving an error message similar to the one shown in the following example.

```
***** Exception at 01128840 : PSII$REMOVE_BOTTOM + 000006B0
%COSI-F-BUGCHECK, internal consistency failure
```

This problem could only occur if the following conditions were met:

- A DROP MODULE ... CASCADE statement had been executed for a module which was referenced by existing modules in the database. In this case the CASCADE was used because the default or explicit RESTRICT option caused an error to be raised.
- The module was later replaced by a revised version.
- This database was subsequently exported using the SQL EXPORT command and a new database was created using the SQL IMPORT command. The problem occurred when using this new database.

When the revised module was created it was given a different creation date and module ID. The SQL EXPORT command used the module ID ordering when exporting the modules and the modules were now exported out of order. In this case, modules which depend on other modules were seen first.

The SQL IMPORT command tried to create the dependency rows when the modules were re-created out of order by modifying the incomplete rows in the RDB$INTERRELATIONS table. However, only the rows were updated and not the index RDB$INTER_OBJ_SUBOBJ_NDX. The bugcheck occurred because the index and table did not contain the same key values.

The only workaround is to avoid the DROP MODULE statement. The existing modules, functions and procedures will continue to work correctly because this index is not used at runtime.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.3.

After Oracle Rdb7 Release 7.0.1.3 has been installed, execute the corrective action shown in the example below. This procedure can be used if you encounter this problem, or if you simply want to verify that there is no problem in your database.

```
SQL> ATTACH 'FILENAME NEW_OINT';
SQL> SET TRANSACTION READ WRITE;
SQL> SET FLAGS 'VALIDATE_ROUTINE,INDEX_STATS';
SQL> CREATE MODULE M_DUMMY
cont>     LANGUAGE sql
cont>     PROCEDURE P_DUMMY;
cont>     BEGIN
cont>     END;
cont> END MODULE;
~Ai: scan/repair RDB$INTERRELATIONS
~Ai: missing key M1                              P1
~Ai: remove old                                  P1
~Ai: missing key M1                              P2
~Ai: remove old                                  P2
SQL> COMMIT; -- make index fix permanent
SQL> DROP MODULE M_DUMMY; -- no longer needed
SQL> COMMIT;
```

If the flag, VALIDATE_ROUTINE, is enabled when a CREATE MODULE statement is executed. Rdb will scan and repair the affected index. The created module (M_DUMMY in this example) can be deleted if it is not needed. If the flag INDEX_STATS is used and no missing keys are reported then the transaction may be rolled back because this indicates that the index correctly reflects the state of the RDB$INTERRELATIONS table. This procedure need only be performed once on each database.

The VALIDATE_ROUTINE flag is used to validate routines and modules in the database and the INDEX_STATS flag generates log messages during the repair process.

If the database is not being used by others then replacing the SET TRANSACTION statement with the following statement could improve execution time because virtual memory usage is reduced and I/O to the snapshot file (.snp) is eliminated.

```
SQL> SET TRANSACTION READ WRITE RESERVING RDB$INTERRELATIONS FOR EXCLUSIVE WRITE;
```

4.3 Oracle RMU Errors Fixed

4.3.1 RMU/REPAIR/SPAM Reversed the Effects of Truncate Table

Bug 636618.

For uniform page-format areas, a space area management (SPAM) page consists of threshold information and a list of logical area identifiers to page-range assignments. This list is referred to as a clump list. When the contents of a table (logical area) are deleted using a SQL Truncate Table command, Oracle Rdb does not go to every data page affected and mark it as deleted. Instead, it marks the clump list entries for pages in the affected table as deleted and leaves the data pages unmodified. If, after a Truncate Table command is performed, an RMU Repair Spam operation is done on the uniform physical area in which the truncated table resides, then the deleted rows could reappear. This was because RMU Repair Spam operation completely rebuilt the clump list on a spam page in addition to recomputing correct threshold values. It rebuilt the clump list by walking through all the data pages in the area file and copying the logical area identifier from the page back into its corresponding clump list entry. It did this without regard for the deleted flag in the clump list entry and thereby reversed the truncate operation on the rows. Although the truncate operation removed any index entries pointing to the deleted rows, after the RMU Repair Spam operation completes, the rows could still be fetched sequentially.

Aside from the obvious error of reversing the work of the SQL Truncate Table command, other problems could occur if pages in the clump list previously marked as deleted were reassigned to other logical areas between the time of the SQL Truncate Table and RMU Repair Spam commands.

This problem has been corrected in Oracle Rdb7 Version 7.0.1.3.

The RMU Repair Spam operation has been corrected to not replace logical area identifiers into clump list entries marked as deleted.

4.3.2 RMU/MOVE_AREA Did Not Delete Moved Files on Failure

Bug 483687.

When the RMU/MOVE_AREA command failed it did not delete the newly created and moved *.rdb, *.rda and *.snp database files. This took up disk space and was confusing because any new database files created and moved up to the moment of failure would be retained along with the original database files before the move command was executed.

The following example shows that even though the RMU/MOVE_AREA command failed, a new jobs.rda file was left in the [.MOVE] directory and a new version of the jobs.snp file was left in the [DEFAULT] directory.

```
$DIR JOBS.*

Directory DISK:[DEFAULT]

JOBS.RDA;1       JOBS.SNP;1

Total of 2 files.
$CREATE/DIR [.MOVE]
$RMU/MOVE_AREA/FILE=DISK:[DEFAULT.MOVE]JOBS.RDA/SNAP=(FILE=JOBS.SNP,-
 ALLOCATION=1000000) MF_PERSONNEL JOBS
%RMU-F-FILACCERR, error extending file DISK:[DEFAULT]JOBS.SNP;1
-SYSTEM-W-DEVICEFULL, device full - allocation failure
%RMU-F-FTL-MOVE, Fatal error for MOVE operation at 20-JUN-1998 11:11:56.98

$DIR JOBS.*

Directory DISK:[DEFAULT]

JOBS.RDA;1       JOBS.SNP;2 JOBS.SNP;1 MOVE.DIR;1

Total of 4 files.

$DIR [.MOVE]

Directory DISK:[DEFAULT.MOVE]

JOBS.RDA;1

Total of 1 files.
```

There is no workaround for this problem.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.3.

Now, if a failure occurs when the RMU/MOVE_AREA command is executed any new *.RDB, *.RDA and *.SNP Rdb database files created by the RMU/MOVE_AREA command are deleted and only the original Rdb database files as they existed before the RMU/MOVE_AREA command are retained.

4.3.3 RMU/VERIFY Reports RMU-W-BADFNMAIJ Warning

Bug 657250.

Performing an RMU/VERIFY command on a database could result in RMU-W-BADFNMAIJ warning messages while verifying the database's after image journal files. This message was emitted when the name of the database root file stored in the after image journal file did not match the actual name of the database root file. In some cases, RMU Verify was indicating real corruption in the after image journal files. However, under certain conditions, the RMU-W-BADFNMAIJ message is incorrect and was emitted when the only difference between the two root filenames was a leading underscore character in the actual root filename's device component. This is shown in the following example.

```
$ RMU/VERIFY MF_PERSONNEL.RDB
%RMU-W-BADFNMAIJ, after-image journal file contains references to wrong database
                  expected: "_$111$DUA369:[RMUWORK]MF_PERSONNEL.RDB;1",
    found: "$111$DUA369:[RMUWORK]MF_PERSONNEL.RDB;1"
```

This situation can occur when an Oracle Rdb 6.0 database with multiple fixed-size after image journal files is converted to release 6.1 or later.

There is no workaround to this problem.

This problem has been corrected in Oracle Rdb7 Version 7.0.1.3.

4.3.4 RMU/SHOW STATISTICS Not Exited Using the $FORCEX System Service

If a process running the RMU/SHOW STATISTICS utility was stalled doing screen I/O due, for example, to the user pressing CONTROL/S (also known as XOFF), it was not possible to exit the process using the $FORCEX system service. When a database shutdown was requested with the /ABORT=FORCEXIT qualifier, processes running the RMU/SHOW STATISTICS utility would not run down and would remain connected to the database.

The occurrences of this problem have been reduced in Oracle Rdb7 Release 7.0.1 .3. The RMU/SHOW STATISTICS utility now executes primarily at non-AST level in user mode. This allows the forcexit request to be delivered more reliably. In addition, an exit handler has been added to help ensure that the database is closed in cases where I/O to the screen is blocked.

4.3.5 RMU/ANALYZE Data Record Count Was Zero for Segmented Strings

Bug 467985.

The RMU Analyze command incorrectly displayed the data record count as zero for the RDB$SEGMENTED_STRINGS logical area residing in a MIXED page format storage area.

The following example demonstrates the error:

```
$ RMU/ANALYZE/AREA=RESUME_LISTS MF_PERSONNEL

  ------------------------------------------------------------------------------

   Storage analysis for storage area: RESUME_LISTS - file:
   DISK1:[RMUWORK]RESUME_LISTS.RDA;1
   Area_id: 9,  Page length: 3072,  Last page: 31

   Bytes free: 92728 (97%), bytes overhead: 2041 (2%)
   Spam count: 1, AIP count: 0, ABM count: 0
   Data records: 12, bytes used: 463 (0%)
     average length: 39, compression ratio: 1.00
     index records: 0, bytes used: 0 (0%)

  ----------------------------------

   Logical area: RDB$SYSTEM_RECORD for storage area : RESUME_LISTS
   Larea id: 54,  Record type: 0, Record length: 215, Not Compressed

   Data records: 0, bytes used: 150 (0%)
  ----------------------------------

   Logical area: RDB$SEGMENTED_STRINGS for storage area : RESUME_LISTS
   Larea id: 56,  Record type: 0, Record length: 155, Not Compressed

   Data records: 0, bytes used: 0 (0%)

  ------------------------------------------------------------------------------
```

The RMU Analyze command now displays the correct data record count:

```
$ RMU/ANALYZE/AREA=RESUME_LISTS MF_PERSONNEL

  ------------------------------------------------------------------------------

   Storage analysis for storage area: RESUME_LISTS - file:
   DISK1:[RMUWORK]RESUME_LISTS.RDA;1
   Area_id: 9,  Page length: 3072,  Last page: 31

   Bytes free: 92728 (97%), bytes overhead: 2041 (2%)
   Spam count: 1, AIP count: 0, ABM count: 0
   Data records: 12, bytes used: 463 (0%)
     average length: 39, compression ratio: 1.00
     index records: 0, bytes used: 0 (0%)

  ----------------------------------

   Logical area: RDB$SYSTEM_RECORD for storage area : RESUME_LISTS
   Larea id: 54,  Record type: 0, Record length: 215, Not Compressed

   Data records: 0, bytes used: 150 (0%)
  ----------------------------------

   Logical area: RDB$SEGMENTED_STRINGS for storage area : RESUME_LISTS
   Larea id: 56,  Record type: 0, Record length: 155, Not Compressed

   Data records: 12, bytes used: 463 (0%)
     average length: 39

  ------------------------------------------------------------------------------
```

There is no workaround to this problem.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.3.

4.3.6 RMU/ANALYZE Command Incorrectly Determined Compression Setting

Bug 640721.

Performing a RMU/ANALYZE command on a database having tables described by storage maps may have resulted in an incorrect determination of the compression setting for those tables. Normally, a table's compression setting is determined by examining the RDB$FLAGS column for that table's entry in the RDB$RELATIONS system table. However, when a table is further described by a storage map, its compression setting is determined by examining the RDB$FLAGS column for that table's entry in the RDB$STORAGE_MAPS system table. In addition, if a table is vertically partitioned, then its entries in the RDB$STORAGE_MAP_AREAS system table must also be consulted in order to make a correct determination of its compression settings. RMU Analyze now correctly makes use of the information contained in these system tables.

The following example shows how RMU/ANALYZE incorrectly determined the compression setting for TAB1 table.

```
SQL> CREATE DATABASE FILE TESTDB CREATE STORAGE AREA AREA1;
SQL>
SQL> CREATE TABLE TAB1 ( COL1 CHAR(5) );
SQL> CREATE TABLE TAB2 ( COL1 CHAR(5) );
SQL>
SQL> CREATE STORAGE MAP TAB1_MAP FOR TAB1 DISABLE COMPRESSION
cont> STORE IN AREA1;
SQL>
SQL> SHOW STORAGE MAPS TAB1_MAP;
       TAB1_MAP
   For Table:  TAB1
   Partitioning is: UPDATABLE
   Store clause: STORE in area1
   Compression is: DISABLED

SQL> INSERT INTO TAB1 VALUES ('AAAAA');
  1 row inserted
SQL> INSERT INTO TAB1 VALUES ('BBBBB');
  1 row inserted
SQL> INSERT INTO TAB1 VALUES ('CCCCC');
  1 row inserted
SQL>
SQL> INSERT INTO TAB2 VALUES ('11111');
  1 row inserted
SQL> INSERT INTO TAB2 VALUES ('22222');
  1 row inserted
SQL> INSERT INTO TAB2 VALUES ('33333');
  1 row inserted
SQL>
SQL> COMMIT;
SQL> EXIT

$ RMU/ANALYZE/EXCLUDE=(SYSTEM,METADATA) TESTDB

   Areas for database - DISK1:[RMUWORK.BUG640721]TESTDB.RDB;1
   Created  5-MAY-1998 13:01:35.38

  ------------------------------------------------------------------------------

   Storage analysis for storage area: RDB$SYSTEM - file:
   DISK1:[RMUWORK]TESTDB.RDA;1
   Area_id: 1,  Page length: 1024,  Last page: 604

   Bytes free: 215103 (35%), bytes overhead: 184409 (30%)
   Spam count: 1, AIP count: 6, ABM count: 141
   Data records: 1135, bytes used: 218984 (35%)
     average length: 193, compression ratio: 1.00
     index records: 117, bytes used: 38066 (6%)
       B-Tree: 30816, Hash: 0, Duplicate: 7250, Overflow: 0

  ----------------------------------

   Logical area: TAB2 for storage area : RDB$SYSTEM
   Larea id: 48,  Record type: 26, Record length: 13, Compressed

   Data records: 3, bytes used: 21 (0%)
     average length: 7, compression ratio:  .87

  ------------------------------------------------------------------------------

   Storage analysis for storage area: AREA1 - file: DISK1:[RMUWORK.BUG640721]AREA1.RDA;1
   Area_id: 2,  Page length: 1024,  Last page: 403

   Bytes free: 391765 (95%), bytes overhead: 20883 (5%)
   Spam count: 1, AIP count: 0, ABM count: 3
   Data records: 3, bytes used: 24 (0%)
     average length: 8, compression ratio:  .12
     index records: 0, bytes used: 0 (0%)

  ----------------------------------
```

```
   Logical area: TAB1 for storage area : AREA1
   Larea id: 49,  Record type: 25, Record length: 13, Compressed

   Data records: 3, bytes used: 24 (0%)
     average length: 8, compression ratio:  .12

  -------------------------------------------------------------------------------
```

The correct analysis for the database is shown below.

```
$ RMU/ANALYZE/EXCLUDE=(SYSTEM,METADATA) TESTDB

   Areas for database - DISK1:[RMUWORK.BUG640721]TESTDB.RDB;1
   Created  5-MAY-1998 12:58:33.99

  -------------------------------------------------------------------------------

   Storage analysis for storage area: RDB$SYSTEM - file:
   DISK1:[RMUWORK]TESTDB.RDA;1
   Area_id: 1,  Page length: 1024,  Last page: 604

   Bytes free: 215103 (35%), bytes overhead: 184409 (30%)
   Spam count: 1, AIP count: 6, ABM count: 141
   Data records: 1135, bytes used: 218984 (35%)
     average length: 193, compression ratio: 1.00
     index records: 117, bytes used: 38066 (6%)
       B-Tree: 30816, Hash: 0, Duplicate: 7250, Overflow: 0

  ----------------------------------

   Logical area: TAB2 for storage area : RDB$SYSTEM
   Larea id: 48,  Record type: 26, Record length: 13, Compressed

   Data records: 3, bytes used: 21 (0%)
     average length: 7, compression ratio:  .87

  -------------------------------------------------------------------------------

   Storage analysis for storage area: AREA1 - file:
   DISK1:[RMUWORK]AREA1.RDA;1
   Area_id: 2,  Page length: 1024,  Last page: 403

   Bytes free: 391765 (95%), bytes overhead: 20883 (5%)
   Spam count: 1, AIP count: 0, ABM count: 3
   Data records: 3, bytes used: 24 (0%)
     average length: 8, compression ratio: 1.00
     index records: 0, bytes used: 0 (0%)

  ----------------------------------

   Logical area: TAB1 for storage area : AREA1
   Larea id: 49,  Record type: 25, Record length: 13, Not Compressed

   Data records: 3, bytes used: 24 (0%)
     average length: 8

  -------------------------------------------------------------------------------
```

There is no workaround to this problem.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.3.

4.3.7 Database Shutdown Message Not Received by RMU/ANALYZE Operation

Bug 349363.

A process performing an RMU/ANALYZE operation did not receive the database shutdown message from a concurrent process issuing an RMU/CLOSE operation using the ABORT qualifier. Note that this problem only occurred when performing a logical area or storage area analysis of a database.

There is no workaround to this problem.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.3.

4.3.8 RMU/SET PRIVILEGE Command Failed with Searchlist Logical Specified

Bug 612157.

Performing a RMU/SET PRIVILEGE command using a searchlist logical name to specify the database root file on the command line resulted in a failure to change the database root access control list as intended. The command produced no error message and simply exited.

When a searchlist logical name is used to specify the database root file on the command line, RMU performs an RMS $PARSE operation using the SYNCHK option in order to release resources. However, in doing so, certain contextual information in the NAMBLK structure is cleared. This information is later used to decide if the source and target root files are the same when using the /LIKE qualifier. Although the /LIKE qualifier was not used in this case, due to the cleared NAMBLK context, the RMU/SET PRIVILEGE command incorrectly assumed that the source and target root files were the same, took no further action and exited.

As a workaround to this problem, do not use a searchlist logical to specify the database root file on the RMU Set Privilege command.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.3.

4.3.9 RMU/UNLOAD Incorrectly Unloaded Data Into a Delimited Text File

Bug 653957.

Performing an RMU/UNLOAD command using the /FORMAT=DELIMITED_ TEXT qualifier resulted in either an access violation, file I/O write error, or incorrect column values being written to the unload file if all columns in a table were unloaded and the table had one or more computed-by or virtual fields defined.

When RMU unloads a table into a delimited text output file, it must dynamically compute a list of column offsets which it uses to search through the data buffer returned by the Rdb database engine. Normally computed-by fields are not unloaded and their column offsets are not considered when building the list. However, RMU was incorrectly factoring in the offsets of computed-by fields, thus corrupting the list.

Each offset in the list indexes into a length/value pair in the data buffer for each column unloaded in a row. Because the offsets were incorrect, arbitrary values were interpreted as columns lengths. This resulted in either an access violation, file I/O write error, or incorrect column values being written to the unload file, depending on the values returned in a given data buffer.

The following example shows what the file I/O write error looks like on an OpenVMS system.

```
$ RMU/UNLOAD/RECORD=(FILE=TAB1,FORMAT=DELIMITED,PREFIX="«",SUFFIX="»",NULL) -
    TESTDB TAB1 TAB1.UNL
  %RMU-E-OUTFILDEL, Fatal error, output file deleted
  -COSI-F-WRITERR, write error
  -RMS-F-RSZ, invalid record size
```

The second example shows a case in which incorrect results could be written to the unload file.

```
SQL> CREATE DATABASE FILENAME 'TESTDB';
SQL>      CREATE TABLE TABLE1 (
cont>       INT_FIELD  INTEGER,
cont>       VIRT_FIELD COMPUTED BY (INT_FIELD + 2),
cont>       CHAR_FIELD CHAR (10));
SQL> INSERT INTO TABLE1(INT_FIELD,CHAR_FIELD) VALUES (1,'0123456789');
  1 row inserted
SQL> INSERT INTO TABLE1(INT_FIELD,CHAR_FIELD) VALUES (2,'1234567890');
  1 row inserted
SQL> INSERT INTO TABLE1(INT_FIELD,CHAR_FIELD) VALUES (3,'2345678901');
  1 row inserted
SQL> INSERT INTO TABLE1(INT_FIELD,CHAR_FIELD) VALUES (4,'3456789012');
  1 row inserted
SQL> INSERT INTO TABLE1(INT_FIELD,CHAR_FIELD) VALUES (5,'4567890123');
  1 row inserted
SQL> COMMIT;
SQL> EXIT;
$
$  ! The values for the char_field column are incorrect.
$
$ RMU/UNLOAD/RECORD=(FILE=UNLOAD1.RRD,FORMAT=DELIMITED) -
  TESTDB TABLE1 UNLOAD1.UNL
  %RMU-I-DATRECUNL,   5 data records unloaded.
$ TYPE UNLOAD1.UNL
  "1",""
  "2",""
  "3",""
  "4",""
  "5",""
$
$  ! By explicitly selecting fields using the /FIELDS qualifier the correct
$  ! values are unloaded.
$
$ RMU/UNLOAD/RECORD=(FILE=UNLOAD2.RRD,FORMAT=DELIMITED) -
  /FIELDS=(INT_FIELD,CHAR_FIELD) TESTDB TABLE1 UNLOAD2.UNL
  %RMU-I-DATRECUNL,   5 data records unloaded.
$ TYPE UNLOAD2.UNL
  "1","0123456789"
  "2","1234567890"
  "3","2345678901"
  "4","3456789012"
  "5","4567890123"
$
$  ! By using the /VIRTUAL qualifier all fields (including the computed-by one)
$  ! are unloaded correctly.
$
$ RMU/UNLOAD/RECORD=(FILE=UNLOAD3.RRD,FORMAT=DELIMITED)/VIRTUAL -
  TESTDB TABLE1 UNLOAD3.UNL
  %RMU-I-DATRECUNL,   5 data records unloaded.
$ TYPE UNLOAD3.UNL
  "1","3","0123456789"
  "2","4","1234567890"
  "3","5","2345678901"
```

```
  "4","6","3456789012"
  "5","7","4567890123"
```

As a workaround to this problem, use either the /FIELDS or the /VIRTUAL qualifier, as shown in the above example.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.3.

4.3.10 RMU Collect Generated Incorrect Prefix Cardinalities

Bug 557641.

In prior releases of Oracle Rdb7, the RMU/COLLECT OPTIMIZER_STATISTICS command would collect incorrect prefix cardinalities for any index which had index compression enabled.

The following example shows the results for three indexes defined on the EMPLOYEES table in the sample PERSONNEL database.

```
$ RMU/COLLECT OPT MF_PERSONNEL-
    /TABLE=EMPLOYEES-
    /STAT=CARDINALITY-
    /LOG
   .
   .
   .
--------------------------------------------------------------------------------

Optimizer Statistics collected for table : EMPLOYEES

  Cardinality           : 100
Index name : EMP3
  Index Cardinality     : 100
  Segment Column                Prefix cardinality
    LAST_NAME                      98
    FIRST_NAME                     100
    EMPLOYEE_ID                    0

Index name : EMP2
  Index Cardinality     : 100
  Segment Column                Prefix cardinality
    LAST_NAME                      83
    FIRST_NAME                     100
    EMPLOYEE_ID                    0

Index name : EMP1
  Index Cardinality     : 100
  Segment Column                Prefix cardinality
    LAST_NAME                      100
    FIRST_NAME                     100
    EMPLOYEE_ID                    0
```

Each of the indexes EMP1, EMP2, and EMP3 have the same structure, but different compression settings (enabled, disabled, and enabled with a minimum run length of 3). The RMU Collect command reported different results for the prefix cardinality for each index. Only EMP2, which has compression disabled, is correct.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.3.

RMU/COLLECT should be run again after the upgrade if it was previously run for any index which has compression enabled. This includes the Rdb system tables if the database was created with the clause SYSTEM INDEX COMPRESSION IS ENABLED, and RMU/COLLECT was run with the /SYSTEM_RELATIONS qualifier. The /INDEX or /TABLE qualifier can be used to limit the number of tables and indexes processed.

4.4 Hot Standby Errors Fixed

4.4.1 LRS Re-Initialized AIJ Journal

When using the Hot Standby feature, it was possible for the Log Replication Server (LRS) to not detect that the AIJ Backup Server (ABS) had already initialized an AIJ journal. This resulted in the LRS re-initializing the new AIJ journal, which is a very time consuming operation for the LRS server to perform.

This problem has been corrected in Oracle Rdb Release 7.0.1.3. The LRS now properly detects when the ABS has previously initialized an AIJ journal.

4.4.2 Hot Standby Database Synchronization Terminated After Ten Minutes

It was possible for the RMU/REPLICATE AFTER_JOURNAL START command to terminate if the database synchronization operation took longer than 10 minutes.

There is no automatic workaround to this problem. The AIJ journals can be backed up and manually recovered on the standby database to reduce the overall database synchronization time to less than 10 minutes.

This problem has been corrected in Oracle Rdb Release 7.0.1.3. The RMU/REPLICATE AFTER_JOURNAL START command now synchronizes the master and standby databases regardless of the overall duration.

4.4.3 Standby Database Inconsistent Following an LRS Server Failure

Following abnormal termination of the Log Replication Server (LRS), it was possible for the standby database to lose replicated data.

This problem only occurred when the LRS was abnormally terminated, such as when using the STOP/ID command. The problem did not occur for network failure or normal replication shutdown.

The effects of the problem can be reduced, but not entirely eliminated, by:

- Using the /ONLINE qualifier at standby replication startup time.
- Reducing the number of buffers allocated to the LRS
- Specifying a high RDM$BIND_CLEAN_BUF_CNT logical name value.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.3. The LRS now properly flushes its buffers completely before each LRS checkpoint interval, thereby ensuring reliable database integrity in the case of abnormal termination.

5
Documentation Corrections

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

5.1 Documentation Corrections

5.1.1 Partition-clause is Optional on CREATE STORAGE MAP

Bug 642158.

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

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

5.1.2 Oracle Rdb Logical Names

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

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

5.1.3 Waiting for Client Lock Message

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

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

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

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

The leftmost value seen in the hexadecimal output contains the id of the object. The id is described below for tables, routines, modules and storage map areas.

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

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

Table 5–1 Object Type Values

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

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

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

```
Process.ID Since......   Stall.reason.......................... Lock.ID.
46001105:2 10:40:46.38 - waiting for client '........' 000000190000000400000055
                                                    ❶       ❷       ❸       ❹
```

The following list describes each part of the client lock:

❶ '........' indicates nonprintable characters

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

❸ 00000004 indicates object type 4 which is a table

❹ 00000055 indicates this is a client lock

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

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

Note

Because the full client lock output is long, it may require more space than is allotted for the Stall.reason column and therefore can be overwritten by the Lock.ID. column output.

For more detailed lock information, perform the following steps:

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

5.1.4 Documentation Error in the Oracle Rdb7 Guide to Database Performance And Tuning

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

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

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

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

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

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

5.1.5 SET FLAGS Option IGNORE_OUTLINE Not Available

Bug 510968.

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

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

5.1.6 SET FLAGS Option INTERNALS Not Described

The *Oracle Rdb7 SQL Reference Manual* does not described the option INTERNALS in table 7-6 in the SET FLAGS section. This keyword was available in first release of Oracle Rdb7 and is used to enable debug flags output for internal queries such as constraints, and triggers. It can be used in conjunction with other options such as STRATEGY, BLR and EXECUTION. For example, the following flags settings are equivalent to defining the RDMS$DEBUG_FLAGS as

"ISn" and shows the strategy used by the triggers actions on the AFTER DELETE trigger on EMPLOYEES.

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

Alias RDB$DBHANDLE:
Flags currently set for Oracle Rdb:
   INTERNALS,STRATEGY,PREFIX,REQUEST_NAMES
SQL> DELETE FROM EMPLOYEES WHERE EMPLOYEE_ID = '00164';
~S: Trigger name  EMPLOYEE_ID_CASCADE_DELETE
Get     Temporary relation      Retrieval by index of relation DEGREES
  Index name  DEG_EMP_ID [1:1]
~S: Trigger name  EMPLOYEE_ID_CASCADE_DELETE
Get     Temporary relation      Retrieval by index of relation JOB_HISTORY
  Index name  JOB_HISTORY_HASH [1:1]
~S: Trigger name  EMPLOYEE_ID_CASCADE_DELETE
Get     Temporary relation      Retrieval by index of relation SALARY_HISTORY
  Index name  SH_EMPLOYEE_ID [1:1]
~S: Trigger name  EMPLOYEE_ID_CASCADE_DELETE
Conjunct        Get     Retrieval by index of relation DEPARTMENTS
  Index name  DEPARTMENTS_INDEX [0:0]
Temporary relation      Get     Retrieval by index of relation EMPLOYEES
  Index name  EMPLOYEES_HASH [1:1]       Direct lookup
1 row deleted
```

5.1.7 Documentation for VALIDATE_ROUTINE Keyword for SET FLAGS

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

This example shows the revalidation of a stored procedure. When the stored routine is successfully prepared (but not executed) the setting of VALIDATE_ ROUTINE causes the entry for this routine in the RDB$ROUTINES system table to set a valid.

```
SQL> SET TRANSACTION READ WRITE;
SQL> SET FLAGS 'VALIDATE_ROUTINE';
SQL> SET NOEXECUTE;
SQL> CALL ADD_EMPLOYEE ('Smith');
SQL> SET EXECUTE;
SQL> COMMIT;
```

In this example the use of the SET NOEXECUTE statement in interactive SQL allows the stored routine to be successfully compiled, but it is not executed.

5.1.8 Documentation for Defining the RDBSERVER Logical Name

Bugs 460611 and 563649.

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

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

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

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

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

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

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

5.1.9 Undocumented SET Commands and Language Options

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

5.1.9.1 QUIET COMMIT Option

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

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

Note

Within a compound statement the COMMIT and ROLLBACK in this case is ignored.

Examples

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

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

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

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

PROCEDURE S_TXN (SQLCODE);
SET TRANSACTION READ WRITE;

PROCEDURE C_TXN (SQLCODE);
COMMIT;
```

5.1.9.2 COMPOUND TRANSACTIONS Option

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

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

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

Examples

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

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

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

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

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

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

5.1.10 Undocumented Size Limit for Indexes with Keys Using Collating Sequences

Bug 586079.

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

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

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

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

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

6

Known Problems and Restrictions

This chapter describes problems, restrictions, and workarounds known to exist in Oracle Rdb7 Release 7.0.1.3.

6.0.1 TRUNCATE TABLE Followed by ROLLBACK or Image Exit May Result In Lost Data for UNIFORM Areas

Bugs 676593 and 694930.

At the completion of the TRUNCATE TABLE statement for a UNIFORM area, Oracle Rdb clears on-disk data structures known as ABM pages for the logical areas which are now empty. This operation is not currently journaled by Oracle Rdb7, thus during rollback recovery, the ABM data structures which were cleared are not fully restored. This may result in some data becoming temporarily inaccessible when processing the table using SEQUENTIAL access on the UNIFORM areas.

The following example demonstrates this behavior:

```
SQL> ATTACH 'FILE EST';
SQL> SHOW TABLE T1
Information for table T1

Columns for table T1:
Column Name                     Data Type        Domain
-----------                     ---------        ------
F1                              INTEGER
F2                              CHAR(100)
F3                              CHAR(100)

Indexes on table T1:
I1                              with column F2
  Duplicates are allowed
  Type is Ranked
    Duplicates are Compressed
  Compression is DISABLED
  Node size  430

SQL> SELECT COUNT(*) FROM T1;

       8793
1 row selected

SQL> TRUNCATE TABLE T1;
SQL> ROLLBACK;
$
$ RMU/VERIFY/ALL EST
%RMU-W-ABMBITERR, inconsistency between spam page 1091 and bit 2 in area bitmap1
%RMU-E-BADABMPAG,       error verifying ABM pages
```

After rollback, the data pages for the truncated logical areas are correctly restored, however, the missing ABMS data structures can cause skipped SPAM intervals during sequential scans. RMU/REPAIR/ABM can be used to restore the ABM data structures cleared during TRUNCATE TABLE.

This problem will be corrected in the next mandatory update to Oracle Rdb7.

6.0.2 Improved Status of Row Cache Feature

Since Oracle Rdb7 Release V7.0 was made available, the row cache feature has gone through considerable refinement and improvement to better handle read/write caches with the integrity and recoverability that our customers expect. Although this release includes improved handling of read/write activity, the full read/write capabilities of row caches are not quite ready. Additional work still needs to be done and, therefore, read-only row caches are still the only officially supported type of row cache allowed.

Although read/write row caches cannot be deployed in your production environment with this release, they are robust enough to begin preliminary testing in your development environment.

It is anticipated that this feature will be fully implemented in Oracle Rdb7 Release V7.0.2.

6.0.3 SELECT Query May Bugcheck with "PSII2SCANGETNEXTBBCDUPLICATE" Error

Bug 683916.

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

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

6.0.4 DBAPack for Windows 3.1 is Deprecated

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

6.0.5 Determining Mode for SQL Non-Stored Procedures

Bug 506464.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

6.0.6 DROP TABLE CASCADE Will Result In %RDB-E-NO_META_UPDATE Error

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

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

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

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

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

The workaround to this problem is to first drop the storage map, and then drop the table using the cascade option. The following example shows the workaround. The SHOW statement indicates that the table, index, and storage map were dropped.

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

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

6.0.7 Bugcheck Dump Files with Exceptions at COSI_CHF_SIGNAL

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

For example, consider the following bugcheck file stack information:

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

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

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

6.0.8 Interruptions Possible Using Multistatement or Stored Procedures

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

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

Session 1:

```
SQL> ATTACH 'FILE MF_PERSONNEL';
SQL> CREATE FUNCTION LIB$WAIT (IN REAL BY REFERENCE)
cont> RETURNS INT;
cont> EXTERNAL NAME LIB$WAIT
cont> LOCATION 'SYS$SHARE:LIBRTL.EXE'
cont> LANGUAGE GENERAL
cont> GENERAL PARAMETER STYLE
cont> VARIANT;
SQL> COMMIT;
SQL> EXIT;
$ SQL
SQL> ATTACH 'FILE MF_PERSONNEL';
SQL> BEGIN
cont> DECLARE :LAST_NAME LAST_NAME_DOM;
cont> DECLARE :WAIT_STATUS INTEGER;
cont> LOOP
cont> SELECT LAST_NAME INTO :LAST_NAME
cont>   FROM EMPLOYEES WHERE EMPLOYEE_ID = '00164';
cont> ROLLBACK;
cont> SET :WAIT_STATUS = LIB$WAIT (5.0);
cont> SET TRANSACTION READ ONLY;
cont> END LOOP;
cont> END;
```

Session 2:

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

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

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

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

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

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

6.0.9 RCS Process Can Bugcheck when Global Buffers Are Enabled

If a database that has row caches enabled is opened automatically at attach time, the RCS process is no longer created immediately by the monitor. RCS process creation is delayed until the first request attempts to access a row cache. If, by this time, enough users have attached to the database to completely consume the global buffer allocation, then the RCS process fails because it cannot get its global buffer allocate set.

To avoid this problem, either have the first user access a row cache immediately or manually open the database using the RMU/OPEN command before anyone attaches to it.

6.0.10 Row Cache Not Allowed on Standby Database While Hot Standby Replication is Active

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

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

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

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

6.0.11 RMU Online Verification Operations and Row Cache

When using row caches, some RMU online verification operations may report errors in the database structure and may not be generally reliable in all verifications. These errors may be due to RMU validating the on-disk database structure and not the actual logical database structure including the row cache contents.

For example, one of the functions that is performed by RMU/VERIFY is to ensure that system records in mixed format areas have a "system record" record ID. However, when a physical row cache is being used, the row on the database page may be marked as "reserved by record cache" because the row has been modified in the record cache but has not yet been written to disk.

In this example, the database ID of *00002011* refers to the "reserved by record cache" record type and *00002001* refers to the system record type:

```
$ RMU/VERIFY/ONLINE DKA0:[DB]MYDB.RDB;1
%RMU-E-PAGSYSREC, area INDEX_MIXED_AREA, page 3
                  system record contains an invalid database ID
                  expected: 00002001 (hex), found: 00002011 (hex)
```

This restriction will be corrected in a future release of Oracle Rdb7.

6.0.12 Adding Row Caches Requires Exclusive Database Access

Adding row caches with the ALTER DATABASE ADD CACHE command now requires exclusive database access.

Previously, it was possible for a new row cache to be added online. This new cache would be seen by users attaching to the database after the cache was created, but users that were already attached to the database would not be able to access the cache and would return results from the database without referencing the cache. This situation resulted in database corruption.

6.0.13 Hot Standby Replication Waits When Starting If Read Only Transactions Running

Hot Standby replication will wait to start if there are read-only (snapshot) transactions running on the standby database. The LRS (Log Rollforward Server) will wait until the read-only transaction(s) commits and then replication will continue.

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

6.0.14 Using the SYS$LIBRARY:SQL_FUNCTIONS70.SQL Oracle Functions Script

All OpenVMS platforms.

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

The following example shows the error:

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

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

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

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

In a multi-version environment, use the setting shown in the following example:

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

6.0.15 DECC and Use of the /STANDARD Switch

Bug 394451.

All OpenVMS platforms.

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

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

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

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

Notice that the /STANDARD switch has other options in addition to RELAXED_ ANSI89 and VAXC. Those are also not supported.

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

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

One factor contributing to Oracle Rdb process page faulting is sorting operations. Common causes of sorts include the SQL GROUP BY, ORDER BY, UNION, and DISTINCT clauses specified for 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

sharable image. Database import and RMU load operations do call the OpenVMS sort run-time library.

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

Sort code does not directly consider the 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 code then uses a value of 75% of the returned working set for virtual memory scratch space. The scratch space is then initialized and the sort begins.

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

When a process's 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 Oracle Rdb sort operations. When the operation can not be done in available memory, sort code will use temporary disk files to hold the data as it is being sorted. The *Oracle Rdb Guide to Performance and Tuning* contains more detailed information about sort work files.

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

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

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

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

6.0.17 Performance Monitor Column Mislabeled

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

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

Bug 521583.

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

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

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

6.0.19 RMU Backup Operations and Tape Drive Types

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

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

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

6.0.20 Use of RDB from Shared Images

All OpenVMS platforms.

Bug 470946.

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

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

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

6.0.21 Interactive SQL Command Line Editor Rejects Eight Bit Characters

Digital UNIX platform.

The interactive SQL command line editor on Digital UNIX can interfere with entering eight bit characters from the command line. The command line editor assumes that a character with the eighth bit set will invoke an editing function. If the command line editor is enabled and a character with the eighth bit set is entered from the command line, the character will not be inserted on the command line. If the character has a corresponding editor function, the function will be invoked; otherwise, the character is considered invalid, and rejected.

There are two ways to enter eight bit characters from the SQL command line; either disable the command line editor or use the command line editor character quoting function to enter each eight bit character. To disable the command line editor, set the configuration parameter RDB_NOLINEDIT in the configuration file.

```
! Disable the interactive SQL command line editor.
RDB_NOLINEDIT   ON
```

To quote a character using the command line editor, type Ctrl/V before each character to be quoted.

6.0.22 Restriction Added for CREATE STORAGE MAP on Table with Data

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

- The storage map must be a simple map which references only the default storage area and represents the current (default) mapping for the table. The default storage area is either RDB$SYSTEM or the area name provided by the CREATE DATABASE ... DEFAULT STORAGE AREA clause.
- The new map may not change THRESHOLDS or COMPRESSION for the table, nor can it use the PLACEMENT VIA INDEX clause. It may only contain one area and may not be vertically partitioned. This new map simply describes the mapping as it exists by default for the table.

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

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

6.0.23 ALTER DOMAIN ... DROP DEFAULT Reports DEFVALUNS Error

Bug 456867.

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

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

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

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

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

6.0.24 Monitor ENQLM Minimum Increased to 32767

All OpenVMS platforms.

In previous versions, the Oracle Rdb7 monitor process (RDMMON) was created with a minimum lock limit (ENQLM) of 8192 locks. This minimum has been increased to 32767 locks (the OpenVMS maximum value).

6.0.25 Oracle Rdb7 Workload Collection Can Stop Hot Standby Replication

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

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

Background Information

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

Note

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

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

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

Workaround

To work around this problem, perform the following:

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

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

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

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

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

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

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

6.0.26 RMU Convert Command and System Tables

When the RMU Convert command converts a database from a previous version to Oracle Rdb7 V7.0 or higher, it sets the RDB$CREATED and RDB$LAST_ ALTERED columns to the timestamp of the convert operation.

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

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

6.0.27 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 will increase a minimum of 13 blocks and a maximum of 597 blocks. The actual increase depends mostly on the maximum number of users specified for the database.

6.0.28 Record Caches and Exclusive Access

If a table has a row-level cache defined for it, the Record Cache Server (RCS) process may acquire a shared lock on the table and prevent any other user from acquiring a protective or exclusive lock on that table.

6.0.29 Strict Partitioning May Scan Extra Partitions

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

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

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

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

6.0.30 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 existing page size and the database has been manually opened or users are active, the add operation fails with the following error:

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

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

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

6.0.31 Restriction on Tape Usage for Digital UNIX V3.2

Digital UNIX platforms.

You can experience a problem where you are unable to use multiple tapes with the Oracle RMU Backup command with Digital UNIX V3.2. Every attempt to recover fails. If this happens and device errors are logged in the system error log, you may have encountered the following situation:

If an error is detected by Digital UNIX during the open operation of the tape device, it is possible that the operation succeeded but the device open reference count is zeroed out. This means that any attempt to use the drive by the process holding the open file descriptor will fail with EINVAL status but another process will be able to open and use the drive even while the first process has it opened.

There is no workaround for this problem. This problem with the magtape driver will be corrected in a future release of Digital UNIX.

6.0.32 Support for Single-File Databases to Be Dropped in a Future Release

Oracle Rdb7 currently supports both single-file and multifile databases on both OpenVMS and Digital UNIX. However, single-file databases will not be supported in a future release of Oracle Rdb7. At that time, Oracle Rdb7 will provide the means to easily convert single-file databases to multifile databases.

Oracle Rdb7 recommends that users with single-file databases perform the following actions:

- Use the Oracle RMU commands, such as Backup and Restore, to make copies, backup, or move single-file databases. Do not use operating system commands to copy, back up, or move databases.
- Create new databases as multifile databases even though single-file databases are supported in Oracle Rdb7 V6.1 and V7.0.

6.0.33 DECdtm Log Stalls

All OpenVMS platforms.

Resource managers using the DECdtm services sometimes suddenly stop being able to commit transactions. The systems have been running fine for some period of time, but suddenly they stop. If Oracle Rdb7 is installed and transactions are being run, an RMU Show command on the affected database will show transactions as being "stalled, waiting to commit".

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

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

```
$ MCR LMCP SET NOTIMEZONE
```

This should force the log to restart.

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

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

This command displays output similar to the following:

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

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

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

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

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

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

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

See your Digital representative for information about patches to DECdtm.

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

All OpenVMS platforms.

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

If you have DECnet/OSI installed on a system running OpenVMS VAX Version 6.1 or higher or OpenVMS Alpha V6.2 or higher, you can run Oracle Rdb7 operations that require the two-phase commit protocol.

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

6.0.35 Multiblock Page Writes May Require Restore Operation

All OpenVMS platforms.

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

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

A future release of Oracle Rdb7 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.

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

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

The problem occurs when a program is connected to a remote database and updates the database, but then just before it commits, the network fails. When the commit executes, SQL shows, as it normally should, that the program has lost the link. Assume that the user waits for a minute or two, then tries the transaction again. The problem is that when the start transaction is issued for the second time, it fails because old information still exists about the previous failed transaction. This occurs even if the user issues a DISCONNECT statement (in V4.1 and earlier, a FINISH statement), which also fails with an RDB-E-IO_ ERROR error message.

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

All OpenVMS platforms.

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

When a copy process completes updates to its respective remote database, it updates the RDB$TRANSFERS system table and then tries to delete any RDB$CHANGES rows not needed by any transfers. During this process, the RDB$CHANGES table cannot be updated by any application process, holding up any database updates until the deletion process is complete. The application stalls while waiting for the RDB$CHANGES table. The resulting contention for RDB$CHANGES SPAM pages and data pages severely impacts performance throughput, requiring user intervention with normal processing.

This is a known restriction in V4.0 and higher. Oracle Rdb7 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 Rdb7 verb is an Oracle Rdb7 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.

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

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

The following example demonstrates this restriction:

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

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

SQL>
```

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

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

6.0.39 ARITH_EXCEPT or Incorrect Results Using LIKE IGNORE CASE

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

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

6.0.40 Different Methods of Limiting Returned Rows From Queries

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

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

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

  Interactive SQL must load its metadata cache for the table before it can process the SELECT statement. In this example, interactive SQL loads its metadata cache to allow it to check that the column EMPLOYEE_ID really exists for the table. The queries on the Oracle Rdb7 system tables RDB$RELATIONS and RDB$RELATION_FIELDS exceed the limit of rows.

  Oracle Rdb7 does not prepare the SELECT statement, let alone execute it. Raising the limit to a number less than 100 (the cardinality of EMPLOYEES) but more than the number of columns in EMPLOYEES (that is, the number of rows to read from the RDB$RELATION_FIELDS system table) is sufficient to read each column definition.

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

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

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

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

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

Keep the differences in mind when limiting returned rows using the logical name RDMS$BIND_QG_REC_LIMIT or the configuration parameter RDB_BIND_QG_REC_LIMIT. They may limit more queries than are obvious. This is important when using 4GL tools, the SQL precompiler, the SQL module processor, and other interfaces that read the Oracle Rdb7 system tables as part of query processing.

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

The CREATE INDEX process involves the following steps:

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

   Because new metadata (which includes the index name) is not written to disk until the end of the index process, Oracle Rdb7 must ensure index name uniqueness across the database during this time by taking a special lock on the provided index name.
3. Read the table for sorting by selected index columns and ordering.
4. Sort the key data.
5. Build the index (includes partitioning across storage areas).
6. Write new metadata to disk.

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

Multiple users can create indexes on the same table by using the RESERVING table_name FOR SHARED DATA DEFINITION clause of the SET TRANSACTION statement. For optimal usage of this capability, Oracle Rdb7 suggests the following guidelines:

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

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

The following table displays the elapsed time benefit when creating multiple indexes in parallel with the SHARED DATA DEFINITION clause. The table shows the elapsed time for 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.

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

Index Create Job	Elapsed Time
Index9	00:01:34.43
Index10	00:01:47.44
All10	00:03:26.66

6.0.42 Side Effect When Calling Stored Routines

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

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

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

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

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

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

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

6.0.43 Nested Correlated Subquery Outer References Incorrect

Outer references from aggregation subqueries contained within nested queries could receive incorrect values, causing the overall query to return incorrect results. The general symptom for an outer query that returned rows 1 to n was that the inner aggregation query would operate with the nth - 1 row data (usually NULL for row 1) when it should have been using the nth row data.

This problem has existed in various forms for all previous versions of Oracle Rdb7, but only appears in V6.1 and later when the inner of the nested queries contains an UPDATE statement.

The following example demonstrates the problem:

```
SQL> ATTACH 'FILENAME SHIPPING';
SQL> SELECT * FROM MANIFEST WHERE VOYAGE_NUM = 4904 OR
cont>                             VOYAGE_NUM = 4909;
  VOYAGE_NUM       EXP_NUM   MATERIAL              TONNAGE
        4904           311   CEDAR                    1200
        4904           311   FIR                       690
        4909           291   IRON ORE                 3000
        4909           350   BAUXITE                  1100
        4909           350   COPPER                   1200
        4909           355   MANGANESE                 550
        4909           355   TIN                       500
7 rows selected
SQL> BEGIN
cont> FOR :A AS EACH ROW OF
cont>  SELECT * FROM VOYAGE V WHERE V.SHIP_NAME = 'SANDRA C.' OR
cont>                              V.SHIP_NAME = 'DAFFODIL' DO
cont>   FOR :B AS EACH ROW OF TABLE CURSOR MODCUR1 FOR
cont>    SELECT * FROM  MANIFEST M WHERE M.VOYAGE_NUM = :A.VOYAGE_NUM DO
cont>     UPDATE MANIFEST
cont>      SET TONNAGE = (SELECT (AVG (M1.EXP_NUM) *3) FROM MANIFEST M1
cont>                     WHERE M1.VOYAGE_NUM = :A.VOYAGE_NUM)
cont>      WHERE CURRENT OF MODCUR1;
cont>   END FOR;
cont> END FOR;
cont> END;
SQL> SELECT * FROM MANIFEST WHERE VOYAGE_NUM = 4904 OR
cont>                             VOYAGE_NUM = 4909;
  VOYAGE_NUM       EXP_NUM   MATERIAL              TONNAGE
        4904           311   CEDAR                    NULL
        4904           311   FIR                      NULL
        4909           291   IRON ORE                  933
        4909           350   BAUXITE                   933
        4909           350   COPPER                    933
        4909           355   MANGANESE                 933
        4909           355   TIN                       933
7 rows selected
```

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

A workaround is to declare a variable of the same type as the outer reference data item, assign the outer reference data into the variable before the inner query that contains the correlated aggregation subquery, and reference the variable in the aggregation subquery. Keep in mind the restriction on the use of local variables in FOR cursor loops described by Section 2.2.7.

For example:

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

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

6.0.44 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 Rdb7 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 Rdb7 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 Rdb7 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 Rdb7 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 Rdb7.

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

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

All OpenVMS platforms.
The SQL statement INCLUDE SQLDA2 is not supported for use with the PL/I precompiler in Oracle Rdb7 V5.0 or higher.

There is no workaround. This problem will be fixed in a future version of Oracle Rdb7.

6.0.46 SQL Pascal Precompiler Processes ARRAY OF RECORD Declarations Incorrectly

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

To avoid the problem, declare the record as a type and then define your array of that type. For example:

```
    main.spa:

        program main (input,output);

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

        begin
        end.

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

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

    good_def.pin

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

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

6.0.47 RMU Parallel Backup Command Not Supported for Use with SLS

All OpenVMS platforms.

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

6.0.48 Oracle RMU Commands Pause During Tape Rewind

Digital UNIX platforms.

For Oracle Rdb7 V6.1 or higher on Digital UNIX, the Oracle RMU Backup and Restore commands pause under certain conditions.

If multiple tape drives are used for RMU Backup or RMU Restore commands and a tape needs to rewind, the Oracle RMU command pauses until the rewind is complete. This is different from behavior on OpenVMS systems where the command continues to write to tape drives that are not rewinding.

There is no workaround for this problem.

6.0.49 TA90 and TA92 Tape Drives Are Not Supported on Digital UNIX

Digital UNIX platforms.

When rewinding or unloading tapes using either TA90 and TA92 drives, Digital UNIX intermittently returns an EIO error, causing the Oracle RMU operation to abort. This problem occurs most often when Oracle RMU accesses multiple tape drives in parallel. However, the problem occurs even with single-tape drive access.

As a result of this problem, Oracle Rdb on Digital UNIX supports neither TA90 nor TA92 tape drives.

6.1 Oracle CDD/Repository Restrictions for Oracle Rdb7

This section describes known problems and restrictions in Oracle CDD/Repository V7.0 and earlier.

6.1.1 Oracle CDD/Repository Compatibility with Oracle Rdb Features

Some Oracle Rdb features are not fully supported by all versions of Oracle CDD/Repository. Table 6–1 shows which versions of Oracle CDD/Repository support Oracle Rdb features and the extent of support.

In Table 6–1, repository support for Oracle Rdb7 features can vary as follows:

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

Table 6–1 Oracle CDD/Repository Compatibility for Oracle Rdb Features

Oracle Rdb Feature	Minimum Version of Oracle Rdb	Minimum Version of Oracle CDD/Repository	Support
CASE, NULLIF, and COALESCE expressions	V6.0	V6.1	Implicit
CAST function	V4.1	V7.0	Explicit
Character data types to support character sets	V4.2	V6.1	Implicit
Collating sequences	V3.1	V6.1	Explicit
Constraints (PRIMARY KEY, UNIQUE, NOT NULL, CHECK, FOREIGN KEY)	V3.1	V5.2	Explicit
CURRENT_DATE, CURRENT_ TIME, and CURRENT_ TIMESTAMP functions	V4.1	V7.0	Explicit
CURRENT_USER, SESSION_ USER, SYSTEM_USER functions	V6.0	V7.0	Explict
Date arithmetic	V4.1	V6.1	Pass-through
DATE ANSI, TIME, TIMESTAMP, and INTERVAL data types	V4.1	V6.1	Explicit
Delimited identifiers	V4.2	V6.1[1]	Explicit
External functions	V6.0	V6.1	Pass-through
External procedures	V7.0	V6.1	Pass-through
EXTRACT, CHAR_LENGTH, and OCTET_LENGTH functions	V4.1	V6.1	Explicit
GRANT/REVOKE privileges	V4.0	V5.0 accepts but does not store information	Pass-through
Indexes	V1.0	V5.2	Explicit
INTEGRATE DOMAIN	V6.1	V6.1	Explicit
INTEGRATE TABLE	V6.1	V6.1	Explicit
Logical area thresholds for storage maps and indexes	V4.1	V5.2	Pass-through
Multinational character set	V3.1	V4.0	Explicit
Multiversion environment (multiple Rdb versions)	V4.1	V5.1	Explicit
NULL keyword	V2.2	V7.0	Explicit
Oracle7 compatibility functions, such as CONCAT, CONVERT, DECODE, and SYSDATE	V7.0	V7.0	Explicit
Outer joins, derived tables	V6.0	V7.0	Pass-through
Query outlines	V6.0	V6.1	Pass-through
Storage map definitions correctly restored	V3.0	V5.1	Explicit

[1]The repository does not preserve the distinction between uppercase and lowercase identifiers. If you use delimited identifiers with Oracle Rdb7, the repository ensures that the record definition does not include objects with names that are duplicates except for case.

(continued on next page)

Table 6–1 (Cont.) Oracle CDD/Repository Compatibility for Oracle Rdb Features

Oracle Rdb Feature	Minimum Version of Oracle Rdb	Minimum Version of Oracle CDD/Repository	Support
Stored functions	V7.0	V6.1	Pass-through
Stored procedures	V6.0	V6.1	Pass-through
SUBSTRING function	V4.0	V7.0 supports all features V5.0 supports all but V4.2 MIA features [2]	Explicit
Temporary tables	V7.0	V6.1	Pass-through
Triggers	V3.1	V5.2	Pass-through
TRUNCATE TABLE	V7.0	V6.1	Pass-through
TRIM and POSITION functions	V6.1	V7.0	Explicit
UPPER, LOWER, TRANSLATE functions	V4.2	V7.0	Explicit
USER function	V2.2	V7.0	Explict

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

6.1.2 Multischema Databases and CDD/Repository

You cannot use multischema databases with CDD/Repository and Oracle Rdb7 V7.0 and earlier. This problem will be corrected in a future release of Oracle Rdb7.

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

OpenVMS VAX platforms.

Oracle Rdb7 provides special Oracle RMU privileges that use the unused portion of the OpenVMS access control list (ACL) to manage access to Oracle RMU operations.

You can use the RMU Set Privilege and RMU Show Privilege commands to set and show the Oracle RMU privileges. The DCL SHOW ACL and DIRECTORY /ACL commands also show the added access control information; however, these tools cannot translate the names defined by Oracle Rdb7.

Note

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

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

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

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

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

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

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

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

Oracle CDD/Repository Releases 5.2 and higher correct this problem. A version of the Oracle CDD/Repository software that corrects this problem and allows new repositories to be created using Oracle Rdb7 is provided on the Oracle Rdb7 kit for use on OpenVMS VAX systems. See Section 6.1.3.1 for details.

6.1.3.1 Installing the Corrected CDDSHR Images

OpenVMS VAX platforms.

Note

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

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

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

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

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

Note

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

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

6.1.3.2 CDD Conversion Procedure

OpenVMS VAX platforms.

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

Note

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

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

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

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

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

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

7

Enhancements in Oracle Rdb7 Release 7.0.1.1

This chapter describes the enhancements that were introduced in Oracle Rdb7 Release 7.0.1.1.

7.1 Enhancements In All Interfaces

7.1.1 Virtual Memory Statistics No Longer Collected

Oracle Rdb no longer collects virtual memory (VM) statistics and the information is no longer included in the RMU/SHOW STATISTICS utility. This information includes the RMU/SHOW STATISTICS items:

GET_VM calls
FREE_VM calls
GET_VM kilobytes
FREE_VM kilobytes
$EXPREG calls

7.1.2 New Logical Name RDMS$CREATE_LAREA_NOLOGGING

This release of Oracle Rdb7 includes a new logical name which will disable journaling to the recovery unit journal (RUJ) and after-image journal (AIJ) during certain CREATE and ALTER operations.

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

This can be a problem when creating or altering a large index, or reorganizing a storage map for a large table. In these cases, table rows, hash buckets, or B-tree nodes are written to the new logical areas and must be journalled to the RUJ and AIJ files. As the DDL operation proceeds, these records might also be re-journalled due to a subsequent update. The amount of I/O to these journals may be extensive due to large amounts of I/O, and the long duration of the transaction may cause the RUJ and AIJ files to grow quite large.

In this release of Oracle Rdb7, this I/O to the recovery and after-image journals can be almost eliminated. The recovery and after-image journals will only contain a special logical operation with no associated data for the CREATE or ALTER operations. The savings during these DDL operations is less journaling I/O and low disk space requirements for these operations on large tables.

Note

Database administrators must be aware of the possible disadvantages of disabling journaling. The trade off is less I/O during the operation versus

more complex recovery procedures. The changes in recovery are discussed below.

Using the RDMS$CREATE_LAREA_NOLOGGING logical name

To disable journaling for these DDL commands the logical name RDMS$CREATE_LAREA_NOLOGGING can be defined for the process which will perform the DDL operations. Once attached to the database all journaling to new logical areas will be disabled until the transaction which created them has been committed or rolled back.

The values accepted for this logical name are:

0—journaling is not disabled. This is the default if the logical name is not defined.

1—create logical area journaling is disabled. For example, on OpenVMS, the logical would be defined as shown in the following example.

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

What happens if I need to recover my database using the after-image journal?

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

```
$ DEFINE/USER RDMS$CREATE_LAREA_NOLOGGING 1
$ SQL$
ATTACH 'FILE DB$:TEST_NOJOURNAL';

CREATE INDEX T_I ON T (A);
%RDB-W-META_WARN, metadata successfully updated with the reported warning
-RDMS-W-DATACMIT, unjournaled changes made; database may not be recoverable
-RDMS-W-DOFULLBCK, full database backup should be done to ensure future
recovery

COMMIT;
```

When an after-image journal contains a record of an unjournaled create or alter and is used to rollforward the AIJ using the RMU/RECOVER command, then the logical area will be marked to indicate that it is incomplete. This can be seen using RMU/DUMP/LAREA=RDB$AIP.

```
                      0003 0037  014F  logical area 55, physical area 3
                             06  0153  area name length 6 bytes
00000000000000000000454C504D4953  0154  area name 'SIMPLE.........'
  00000000000000000000000000000000  0164  area name '................'
                       00000000  0173  snaps enabled TSN 0
                           000D  0177  record length 13 bytes
                       00000000  0179  MBZ '....'
                             0D  017D  entry is in use
                           0000  017E  MBZ '..'
                         000000  0180  thresholds are (0,0,0)
                             00  0183  MBZ '.'
                                 area is corrupt and cannot be accessed
```

Oracle Rdb will report that the table or index is incomplete (has had unjournaled changes) if an attempt is made to use that table, or index after the recovery is complete. The only option at this time is to drop the table, storage map, or index and repeat the operation. In the following example the index T_I was created with logging disabled, this shows the results after the RMU/RECOVER command was used to recover the database.

```
SQL> SET FLAGS 'STRATEGY';
SQL> -- select using the incomplete index
SQL> -- FAILS
SQL> SELECT * FROM T WHERE A > 0;
Leaf#01 FFirst T Card=5
  BgrNdx1 T_I [1:0] Fan=17
%RDMS-F-DATATBLCMIT, unjournaled changes made to user-defined object
SQL>
SQL> -- now do a sequential scan
SQL> -- SUCCEEDS
SQL> SELECT B FROM T;
Get     Retrieval sequentially of relation T
           B
        NULL
           1
        NULL
           1
           2
5 rows selected
SQL>
SQL> -- now drop the index
SQL> -- SUCCEEDS
SQL> DROP INDEX T_I;
Firstn  Get     Retrieval by index of relation RDB$INDICES
  Index name  RDB$NDX_NDX_NAME_NDX [1:1]         Direct lookup
SQL>
SQL> -- select again (uses sequential scan)
SQL> -- SUCCEEDS
SQL> SELECT * FROM t WHERE A > 0;
Conjunct        Get     Retrieval sequentially of relation T
           A             B
           1          NULL
           1             1
           2          NULL
           2             1
           2             2
5 rows selected
SQL>
SQL> -- recreate the index
SQL> CREATE INDEX T_I ON T (A);
SQL>
SQL> -- select using the new index
SQL> -- SUCCEEDS
SQL> SELECT * FROM t WHERE A > 0;
Leaf#01 FFirst T Card=5
  BgrNdx1 T_I [1:0] Fan=17
           A             B
           1          NULL
           1             1
           2          NULL
           2             1
           2             2
5 rows selected
SQL>
SQL> COMMIT;
```

Note

If CREATE INDEX or ALTER INDEX refers to a HASHED index then this operation also requires updates to the RDB$SYSTEM_RECORD on each page of a MIXED format area. When these records are updated, the changes are always journalled to the recovery and after-image journals. Therefore, some journaling activity will result from these operations.

What if after-image journaling is disabled?

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

What does RMU VERIFY report if one of the logical areas is marked incomplete?

RMU Verify will attempt to ready the incomplete logical area. The following example shows that the index T_I is incomplete (the warning message DATATBLCMIT) and the verify of the B-tree is abandoned.

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

What happens if the CREATE or ALTER statement fails when journaling is disabled?

In this case, the creation of the logical area (which is always journalled) is rolled back. All the data written to that logical area is then erased from the database. To erase the data from a MIXED format area requires that each page of the storage area be processed. This will, most likely, be slower than similar recovery when journaling is enabled.

When recovery is performed using the RMU/RECOVER command, any rolled-back transaction is discarded (not applied to the backup database) and no reference to the incomplete logical area will be encountered.

What if an error occurs during the transaction?

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

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

Although the COMMIT command can be used and the logical area deleted (using a DROP statement), this action may leave data anomalies which couldn't be rolled back at the time the error was detected. Oracle recommends that the transaction be rolled back.

Oracle recommends that the transaction be committed immediately after the CREATE or ALTER statement has successfully completed and avoid performing additional DML statements such as INSERT, UPDATE and DELETE. Committing promptly will avoid the problems described in this section and will also release locks on rows and other database system resources.

What happens if Hot Standby is active on the database?

The Hot Standby Option requires all operations to be journalled, therefore this logical name is ignored for any database enabled for Hot Standby.

Restriction for LIST STORAGE MAP

The disabling of logging is not supported when creating or altering a LIST STORAGE MAP. Please ensure that the RDMS$CREATE_LAREA_NOLOGGING logical name is not defined when adding or making changes to a LIST STORAGE MAP. This also includes performing IMPORT operations which might implicitly create a LIST STORAGE MAP.

The reason for this restriction is that it is not possible for the rollback processing to distinguish between old and new LIST segments which might exist in the storage map. In Release 7.0.1.2 of Oracle Rdb7, the RDMS$CREATE_LAREA_ NOLOGGING logical name will be ignored during create or alter of a LIST STORAGE MAP.

7.1.3 Online Creation of Storage Areas Performed In Parallel

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

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

```
SQL> ATTACH 'FILENAME MYDB';
SQL> ALTER DATABASE FILE MYDB
  ADD STORAGE AREA S1 FILENAME D1:[DB]S1 ALLOCATION 1000000
      SNAPSHOT FILENAME D6:[DB]S1 SNAPSHOT ALLOCATION 10000
  ADD STORAGE AREA S2 FILENAME D2:[DB]S2 ALLOCATION 1000000
      SNAPSHOT FILENAME D7:[DB]S2 SNAPSHOT ALLOCATION 10000
  ADD STORAGE AREA S3 FILENAME D3:[DB]S3 ALLOCATION 1000000
      SNAPSHOT FILENAME D8:[DB]S3 SNAPSHOT ALLOCATION 10000
  ADD STORAGE AREA S4 FILENAME D4:[DB]S4 ALLOCATION 1000000
      SNAPSHOT FILENAME D9:[DB]S4 SNAPSHOT ALLOCATION 10000
  ADD STORAGE AREA S5 FILENAME D5:[DB]S5 ALLOCATION 1000000
      SNAPSHOT FILENAME D0:[DB]S5 SNAPSHOT ALLOCATION 10000;
```

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

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

On OpenVMS, Oracle Rdb attempts to limit the number of parallel operations based on the process's remaining FILLM, ASTLM and DIOLM quotas. To ensure the highest level of performance, the recommended minimums for these process and system quotas for online area additions are listed in Table 7–1, Recommended Quota Minimums.

Table 7–1 Recommended Quota Minimums

Quota	Recommended Minimum
ASTLM	2 times the number of area files being added (including the snapshot storage area files), or 35, whichever is less.
DIOLM	2 times the number of area files being added (including the snapshot storage area files) or 35, whichever is less.
FILLM	At least enough available to open the additional number of storage area files being added (including the snapshot storage area files).
CHANNELCNT	At least enough available to open the additional number of storage area files being added (including the snapshot storage area files).
WSQUOTA	Large enough to avoid excessive page faulting. Each storage area being initialized in parallel requires at least an additional 400 working set pages on a VAX system or 25 working set pages on an Alpha system.

In general, utilizing more disk devices will result in increased performance when adding multiple storage areas. If you specify a large number of storage areas and many areas share the same device, a large multithread count could possibly cause excessive disk head movement, which may result in the storage area creation taking longer than if the areas were created one at a time. If this situation is the case, specify multiple ALTER DATABASE...ADD STORAGE AREA statements

or specify a smaller multithread count using the logical name RDM$BIND_ONLINE_AREA_ADD_MULTITHREAD_COUNT.

7.2 SQL Interface Enhancements

7.2.1 Oracle7 Outer Join Syntax Support

Use of Oracle7 outer join syntax was not supported. Client applications originally written for Oracle7 might have used that syntax and failed. Now they should succeed.

The following example shows the Oracle7 outer join syntax.

```
SELECT * FROM A,B WHERE A.ACOL1(+) = B.BCOL1;
```

7.3 Oracle RMU Enhancements

7.3.1 RMU/SHOW STATISTIC "Transaction Recovery Duration Estimate" Screen

One of the most difficult database attributes to determine is how long the database will be frozen if a process prematurely terminates, or how long a transaction rollback will take. Transaction recovery is affected by many factors, most of which are difficult to determine from runtime information available from the RMU/SHOW STATISTIC utility.

Therefore, the RMU/SHOW STATISTIC utility has been enhanced to provide an *estimate* of the time it will take to rollback a transaction, or to completely recover a failed process.

Disclaimer!

The information provided on the Transaction Recovery Duration Estimate screen is an estimate based on previous process recovery operations and other factors such as page contention and disk throughput.

However, it cannot be stressed enough that this information is an estimate only; the actual process recovery duration may be more or less than described on this screen.

Individual process failure recovery performance can vary widely depending on many factors which cannot be accounted for in the displayed estimate. These factors include lock deadlock stalls, network delays, disk contention and many other system factors such as lock remastering, etc.

The following example provides a sample transaction recovery scenario to consider:

```
Node: ALPH   (1/1/2)    Oracle Rdb X7.0-00 Perf. Monitor 17-AUG-1997 08:50:48.41
Rate: 1.00 Second    Transaction Recovery Duration Estimate Elapsed: 00:29:04.34
Page: 1 of 1            DISK$:[WORK]MF_PERSONNEL.RDB;1     Mode: Online
--------------------------------------------------------------------------------
Process.ID  RUJ.Sz Tx.Rollback DBR.Tx.Undo AIJ.Ckpt Pnd DBR.Tx.Redo DB.Freeze.Tm
2AA0ECC3:1     431 00:00:08.62 00:00:02.15 0:638    222 00:00:00.22 00:00:10.11



--------------------------------------------------------------------------------
Exit Help Menu >next_page <prev_page Refresh Set_rate Write Zoom !
```

The Transaction Recovery Duration Estimate screen provides the following information:

- **Process.ID** - This is the process identifier of a process that has the potential to rollback a transaction or require transaction recovery in the event of process failure.
- **RUJ.Sz** - This is the number of blocks of RUJ information that have been written by the process.
- **Tx.Rollback** - This is the *estimate* of the time it would require for the process to rollback the transaction. Note that this is different from the time it would take the DBR process to rollback the transaction.
- **DBR.Tx.Undo** - This is the *estimate* of the time it would require for the DBR process to "undo" the transaction. The DBR transaction undo duration is typically less than it takes the process to rollback the transaction, due to various optimizations and simplifications in the DBR recovery algorithm.
- **AIJ.Ckpt** - If the fast commit feature is enabled, this is the most recent checkpoint location in the AIJ journal for the process.
- **Pnd** - If AIJ journaling is enabled, this is the number of blocks of AIJ information that has been submitted (pending) but not yet written to the AIJ journal.
- **DBR.Tx.Redo** - If the fast commit feature is enabled, this is the *estimate* of the time it would take the DBR process to redo the failed process' previously committed transactions to the database.
- **DB.Freeze.Tm** - This is the "estimate" of the total time the database would be frozen if the current process were to prematurely terminate.

In the above example, there are three estimates of essential information:

1. Process transaction rollback duration
2. DBR transaction undo and redo duration
3. Total database freeze duration

In the above example, if the process were to rollback the current transaction, it is estimated to take approximately 8 seconds. If the process were to fail prematurely, it is estimated to take the DBR process approximately 2 seconds to undo the transaction, but approximately .25 seconds to redo all previously committed transactions for that process. The total database freeze time is estimated to be approximately 10 seconds.

Validating the screen information can be performed by examining the end of the DBR logfile, which is enabled using the RDM$BIND_DBR_LOG_FILE logical. For example:

```
18-AUG-1997 11:16:31.22 - TSN 0:291 was rolled back
18-AUG-1997 11:16:31.26 - Total recovery duration 10.10 seconds
```

Examining the past history of recovery operations can be performed using the RMU/DUMP/HEADER utility and reviewing the Database Recovery section. For example:

```
    Database Recovery...
      - 2 process failures have occurred (last 18-AUG-1997 11:16:31.26)
      - DBR freeze averaging 5.470 seconds per recovery
        Transaction REDO averaging 0.890 seconds per recovery
        Transaction UNDO averaging 3.465 seconds per recovery
        AIJ recovery averaging 1.10 seconds per recovery
        Global buffer recovery averaging 0.0 seconds per recovery
        Global buffer tx recovery averaging 0.0 seconds per recovery
        Record cache recovery averaging 0.0 seconds per recovery
      - DBR redo averaging 318 AIJ blocks per recovery
      - DBR redo recovery rate averaging 2ms per AIJ block
      - DBR undo averaging 635 RUJ blocks per recovery
      - DBR undo recovery rate averaging 5ms per RUJ block
      - DBR AIJ scan averaging 63 AIJ blocks per recovery
      - DBR AIJ scan rate averaging 1ms per AIJ block
      - Database is consistent but has been modified
      - Full AIJ roll-forward is no longer permitted to this database
   By-Area and By-Page AIJ roll-forward is permitted
      - Full AIJ roll-forward to a newly restored database is permitted
      - Next AIJ sequence number expected is 1
      - Last commit transaction TSN is 0:320
      - AIJ roll-forward is no-quiet-point enabled
```

The Transaction Recovery Duration Estimate screen is only available during online statistics collection. It is not available during binary file replay.

The configuration variable RECOVERY_SORT can be used to sort the Transaction Recovery Duration Estimate screen, by specifying one of the following keywords:

LONGEST_TRANSACTION - Sort by longest transaction rollback duration
LONGEST_UNDO - Sort by longest DBR undo duration estimate
LONGEST_REDO - Sort by longest DBR redo duration estimate
LONGEST_FREEZE - Sort by longest database freeze duration estimate

Of course, these sort criteria can also be selected online using the Config onscreen-menu option.

7.3.2 RMU/SHOW STATISTIC "File Overview" Sorting and Filtering Enhancements

The RMU/SHOW STATISTIC utility File IO Overview and File Lock Overview screens have been enhanced to provide additional sorting and filtering capabilities.

Two new sort options have been added to the screen configuration options, obtained using the Config onscreen-menu. The new Sort Alphabetically option sorts the storage area names without regards to storage area type (data or snapshot). The new Sort Alphabetically by Type option sorts the storage area names within storage area type (data or snapshot).

For example, the following File IO Overview screen shows the standard unsorted display:

```
Node: ALPH   (1/1/2)    Oracle Rdb X7.0-00 Perf. Monitor 25-AUG-1997 09:20:34.19
Rate: 1.00 Second    File IO Overview (Unsorted total I/O)  Elapsed: 00:04:07.54
Page: 1 of 1             DISK$:[WORK]MF_PERSONNEL.RDB;1     Mode: Online
--------------------------------------------------------------------------------
File/Storage.Area.Name..... Sync.Reads SyncWrites AsyncReads AsyncWrits PgCkd
Database Root                      17          0          0          1     0
AIJ (After-Image Journal)           0          0          0          0     0
RUJ (Recovery-Unit Journal)         0          0          0          0     0
ACE (AIJ Cache Electronic)          0          0          0          0     0
All data/snap files                 3          0          0          0     0
data JOBS                           0          0          0          0     0
data MF_PERS_DEFAULT                3          0          0          0     0
data SALARY_HISTORY                 0          0          0          0     0
data DEPARTMENTS                    0          0          0          0     0
data EMPIDS_LOW                     0          0          0          0     0
data EMPIDS_MID                     0          0          0          0     0
data EMPIDS_OVER                    0          0          0          0     0
data EMP_INFO                       0          0          0          0     0
data MF_PERS_SEGSTR                 0          0          0          0     0
snap JOBS                           0          0          0          0     0
snap MF_PERS_DEFAULT                0          0          0          0     0
snap SALARY_HISTORY                 0          0          0          0     0
snap DEPARTMENTS                    0          0          0          0     0
snap EMPIDS_LOW                     0          0          0          0     0
snap EMPIDS_MID                     0          0          0          0     0
snap EMPIDS_OVER                    0          0          0          0     0
snap EMP_INFO                       0          0          0          0     0
snap MF_PERS_SEGSTR                 0          0          0          0     0

--------------------------------------------------------------------------------
Config Exit Filter Help Menu >next_page <prev_page Options Reset Set_rate Write
```

The following File IO Overview screen shows the display sorted alphabetically:

```
Node: ALPH   (1/1/2)    Oracle Rdb X7.0-00 Perf. Monitor 25-AUG-1997 09:20:40.92
Rate: 1.00 Second       File IO Overview (Alphabetical)     Elapsed: 00:04:14.27
Page: 1 of 1             DISK$:[WORK]MF_PERSONNEL.RDB;1     Mode: Online
-------------------------------------------------------------------------------
File/Storage.Area.Name..... Sync.Reads SyncWrites AsyncReads AsyncWrits PgCkd
ACE (AIJ Cache Electronic)            0          0          0          0     0
AIJ (After-Image Journal)             0          0          0          0     0
All data/snap files                   3          0          0          0     0
data DEPARTMENTS                      0          0          0          0     0
snap DEPARTMENTS                      0          0          0          0     0
Database Root                        17          0          0          1     0
data EMPIDS_LOW                       0          0          0          0     0
snap EMPIDS_LOW                       0          0          0          0     0
data EMPIDS_MID                       0          0          0          0     0
snap EMPIDS_MID                       0          0          0          0     0
data EMPIDS_OVER                      0          0          0          0     0
snap EMPIDS_OVER                      0          0          0          0     0
data EMP_INFO                         0          0          0          0     0
snap EMP_INFO                         0          0          0          0     0
data JOBS                             0          0          0          0     0
snap JOBS                             0          0          0          0     0
data MF_PERS_DEFAULT                  3          0          0          0     0
snap MF_PERS_DEFAULT                  0          0          0          0     0
data MF_PERS_SEGSTR                   0          0          0          0     0
snap MF_PERS_SEGSTR                   0          0          0          0     0
RUJ (Recovery-Unit Journal)           0          0          0          0     0
data SALARY_HISTORY                   0          0          0          0     0
snap SALARY_HISTORY                   0          0          0          0     0

-------------------------------------------------------------------------------
Config Exit Filter Help Menu >next_page <prev_page Options Reset Set_rate Write
```

The following File IO Overview screen shows the display sorted alphabetically by type:

```
Node: ALPH   (1/1/2)    Oracle Rdb X7.0-00 Perf. Monitor 25-AUG-1997 09:20:44.42
Rate: 1.00 Second       File IO Overview (Alphabetical)     Elapsed: 00:04:17.77
Page: 1 of 1             DISK$:[WORK]MF_PERSONNEL.RDB;1     Mode: Online
-------------------------------------------------------------------------------
File/Storage.Area.Name..... Sync.Reads SyncWrites AsyncReads AsyncWrits PgCkd
ACE (AIJ Cache Electronic)            0          0          0          0     0
AIJ (After-Image Journal)             0          0          0          0     0
All data/snap files                   3          0          0          0     0
Database Root                        17          0          0          1     0
RUJ (Recovery-Unit Journal)           0          0          0          0     0
data DEPARTMENTS                      0          0          0          0     0
data EMPIDS_LOW                       0          0          0          0     0
data EMPIDS_MID                       0          0          0          0     0
data EMPIDS_OVER                      0          0          0          0     0
data EMP_INFO                         0          0          0          0     0
data JOBS                             0          0          0          0     0
data MF_PERS_DEFAULT                  3          0          0          0     0
data MF_PERS_SEGSTR                   0          0          0          0     0
data SALARY_HISTORY                   0          0          0          0     0
snap DEPARTMENTS                      0          0          0          0     0
snap EMPIDS_LOW                       0          0          0          0     0
snap EMPIDS_MID                       0          0          0          0     0
snap EMPIDS_OVER                      0          0          0          0     0
snap EMP_INFO                         0          0          0          0     0
snap JOBS                             0          0          0          0     0
snap MF_PERS_DEFAULT                  0          0          0          0     0
snap MF_PERS_SEGSTR                   0          0          0          0     0
snap SALARY_HISTORY                   0          0          0          0     0

-------------------------------------------------------------------------------
Config Exit Filter Help Menu >next_page <prev_page Options Reset Set_rate Write
```

Also, a new Filter onscreen-menu option has been added. The Filter onscreen-menu option prompts the user to enter a pattern string that includes wildcard characters. Using wildcard characters in the search pattern, for example, it is possible to find all EMP storage areas using the search pattern "*EMP*".

Note

Search patterns specified without wildcard characters will find exact matches only. For example, the wildcard name "EMP" will find the single storage area whose name is "EMP".

The pattern string may contain either one or both of the two wildcard characters, asterisk (*) and percent (%). The asterisk character is mapped to zero or more characters. The percent character is mapped to only one character.

You may enter a different filter for each screen.

Of course, filtering of the alphabetically sorted storage areas is permitted.

When a filter has been specified, the Filter onscreen-menu option will be highlighted. Selecting the Filter onscreen-menu option and pressing the RETURN key will delete any previously existing filter.

The following example shows the File IO Overview screen filtered using the pattern "*EMP*":

```
Node: ALPH   (1/1/2)    Oracle Rdb X7.0-00 Perf. Monitor 25-AUG-1997 09:25:50.61
Rate: 1.00 Second    File IO Overview (Unsorted total I/O)  Elapsed: 00:00:05.57
Page: 1 of 1             DISK$:[WORK]MF_PERSONNEL.RDB;1     Mode: Online
------------------------------------------------------------------------------
File/Storage.Area.Name........ Sync.Reads SyncWrites AsyncReads AsyncWrits PgCkd
data EMPIDS_LOW                         0          0          0          0     0
data EMPIDS_MID                         0          0          0          0     0
data EMPIDS_OVER                        0          0          0          0     0
data EMP_INFO                           0          0          0          0     0
snap EMPIDS_LOW                         0          0          0          0     0
snap EMPIDS_MID                         0          0          0          0     0
snap EMPIDS_OVER                        0          0          0          0     0
snap EMP_INFO                           0          0          0          0     0


------------------------------------------------------------------------------
Config Exit Filter Help Menu >next_page <prev_page Options Reset Set_rate Write
```

Note

The "data" and "snap" prefixes are *not* part of the storage area name and are not considered when applying a specified filter. For example, the pattern "data*" will **NOT** find all data storage areas.

To control the selection of storage area types, three new sort options have been added to the screen configuration options, obtained using the Config onscreen-menu. The new Display All Storage Areas option displays all storage areas, as has previously been the case. The new Display Data Storage Areas Only option displays only live data storage areas. The new Display Snap Storage Areas Only option displays only snapshot storage areas.

The following example shows the File IO Overview screen displaying only live storage areas:

```
Node: ALPH   (1/1/2)    Oracle Rdb X7.0-00 Perf. Monitor 25-AUG-1997 13:46:22.48
Rate: 1.00 Second    File IO Overview (Unsorted total I/O)  Elapsed: 00:01:46.60
Page: 1 of 1             DISK$:[WORK]MF_PERSONNEL.RDB;1     Mode: Online
-------------------------------------------------------------------------------
File/Storage.Area.Name........ Sync.Reads SyncWrites AsyncReads AsyncWrits PgCkd
data MF_PERS_DEFAULT                   3          0          0          0     0
data DEPARTMENTS                       0          0          0          0     0
data EMPIDS_LOW                        0          0          0          0     0
data EMPIDS_MID                        0          0          0          0     0
data EMPIDS_OVER                       0          0          0          0     0
data EMP_INFO                          0          0          0          0     0
data JOBS                              0          0          0          0     0
data MF_PERS_SEGSTR                    0          0          0          0     0
data SALARY_HISTORY                    0          0          0          0     0


-------------------------------------------------------------------------------
Config Exit Filter Help Menu >next_page <prev_page Options Reset Set_rate Write
```

The following example shows the File IO Overview screen displaying only snapshot storage areas:

```
Node: ALPH   (1/1/2)    Oracle Rdb X7.0-00 Perf. Monitor 25-AUG-1997 13:46:26.06
Rate: 1.00 Second    File IO Overview (Unsorted total I/O)  Elapsed: 00:01:50.18
Page: 1 of 1             DISK$:[WORK]MF_PERSONNEL.RDB;1     Mode: Online
-------------------------------------------------------------------------------
File/Storage.Area.Name........ Sync.Reads SyncWrites AsyncReads AsyncWrits PgCkd
snap MF_PERS_DEFAULT                   0          0          0          0     0
snap DEPARTMENTS                       0          0          0          0     0
snap EMPIDS_LOW                        0          0          0          0     0
snap EMPIDS_MID                        0          0          0          0     0
snap EMPIDS_OVER                       0          0          0          0     0
snap EMP_INFO                          0          0          0          0     0
snap JOBS                              0          0          0          0     0
snap MF_PERS_SEGSTR                    0          0          0          0     0
snap SALARY_HISTORY                    0          0          0          0     0

-------------------------------------------------------------------------------
Config Exit Filter Help Menu >next_page <prev_page Options Reset Set_rate Write
```

7.3.3 RMU/SHOW STATISTIC Utility /OPTION=CONFIRM Qualifier

A new command qualifier has been added to the RMU/SHOW STATISTIC utility: /OPTION=CONFIRM. The CONFIRM keyword indicates that you wish to confirm before exiting from the utility.

This qualifier can also be specified in the configuration file using the CONFIRM_EXIT variable. A value of TRUE indicates you wish to confirm before exiting the utility, while a value of FALSE, the default value, indicates you do *not* want to confirm before exiting the utility.

7.3.4 RMU/SHOW STATISTIC Utility Fast Incremental Backup Display

The RMU/SHOW STATISTIC utility has been enhanced to display fast incremental backup runtime statistics in the Fast Incr Backup Statistics screen, located in the Journaling Information sub-menu.

The following is an example of the Fast Incr Backup Statistics screen:

```
Node: ALPH   (1/1/2)    Oracle Rdb X7.0-00 Perf. Monitor 11-SEP-1997 13:45:05.69
Rate: 0.50 Seconds        Fast Incr Backup Statistics       Elapsed: 00:35:38.17
Page: 1 of 1            DISK$:[WORK]MF_PERSONNEL.RDB;1      Mode: Online
--------------------------------------------------------------------------------

statistic.........      rate.per.second.............. total....... average......
name..............      max..... cur..... avg....... count....... per.trans....

FIB update attempt            32        0       10.3        22033           1.6
FIB map updated                0        0        0.0           15           0.0
SPAM page updated              0        0        0.0           15           0.0
SPAM updt deferred            32        0       10.2        22015           1.6

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

The following explains the statistical information displayed:

- **FIB update attempt**: This statistic indicates the number of times the fast incremental backup (FIB) update operation was attempted. The *attempt* does not always result in the SPAM page being updated.
- **FIB map updated**: This statistic indicates the number of times the FIB map, a per-process data structure, was updated. This data structure indicates when each process no longer needs to update a particular SPAM page any longer.
- **SPAM page updated**: This statistic indicates the number of times a SPAM page was immediately modified to indicate that one or more pages in the SPAM interval have been modified since the last incremental backup. Each SPAM page update results in one synchronous read I/O and one synchronous write I/O operation.
- **SPAM updt deferred**: This statistic indicates the number of times a SPAM page did not need to be immediately modified, but might have to be modified at a later time. In most cases, this statistic closely follows the **FIB update attempt** statistic.

This screen is available during replay of a binary input file, and is also available cluster-wide.

7.3.5 RMU/SHOW STATISTIC Utility "Page Information" Zoom Screen

The RMU/SHOW STATISTIC utility has been integrated with the RMU/DUMP utility to provide runtime database page information displayed on a “zoom” screen. The page information is presented on a “zoom” screen in a format similar to that displayed by the RMU/DUMP/AREA=parea/START=pno/END=pno utility.

The page information zoom screen is currently available from the Stall Messages, Active User Stall Messages, DBR Activity and DBKEY Information screens.

The page information is selected using the PageInfo onscreen-menu option, by pressing the P key.

You will be prompted to select a process from the list of available processes with DBKEY information displayed on the screen. Only those processes displaying physical DBKEY information can be selected.

If the process you selected is accessing a range of pages, you will be prompted to select the desired page from a sub-menu provided.

If you are displaying page information from the DBKEY Information page, you will be prompted to select one of the types of pages being accessed by that process.

It is also possible to display an arbitrary page using the Tools menu, obtained using the exclamation point (!). Select the Display Page Information option and you will be prompted for the desired storage area and page number.

The following caveats apply to the page information display:

- For security reasons, the contents of individual lines on a data page *cannot* be displayed. The contents of area inventory pages cannot displayed, either. Contact your DBA for other methods to display the contents of selected rows.
- Because the page information can be quite lengthy, you are able to migrate through the various pages using the "right-arrow" and "left-arrow" keys (1 page at a time) or the "up-arrow" and "down-arrow" keys (1 line at a time).
- Of course, the page information "zoom" screen contents can be written to disk using the Write onscreen-menu option (W key).
- The PageInfo onscreen-menu option is not available during replay of a binary input file.
- No locking of the selected page actually occurs. Therefore, it may be possible (but unlikely) to display inconsistent page information.

The PageInfo onscreen-menu option identifies and resolves logical DBKEYs and retrieves the corresponding physical DBKEYs.

Note

When using ALG, logical-DBKEYs such as "59:1:-3" are not resolveable, so SHOW STATISTICS retrieves the identified page, which in some cases is not always correct. In the above example, page "1" is a SPAM page, which obviously cannot be the target of the logical DBKEY.

The following is an example of a live data page information display:

```
+-------------------------------------------------------------------------+
|                    0001 00000005  0000  page 5, physical area 1 (data)  |
|                         8EE86A99  0006  checksum = 8EE86A99             |
|                009BA463 0DA1FC74  000A  time stamp = 14-SEP-1997 06:51:12.75 |
|                         0000 006A  0012  106 free bytes, 0 locked       |
|                              0002  0016  2 lines                        |
|                         01AE 0240  0018  line 0: offset 0240, 430 bytes |
|                         01AE 0092  001C  line 1: offset 0092, 430 bytes |
|                          0000018C  0020  line 0: TSN 396                |
|                          000001BF  0024  line 1: TSN 447                |
|                          00000024  03EE  snap page pointer 36           |
|                          000001BF  03F2  snap pointer TSN 447           |
|                              003B  03F6  logical area 59                |
|                          0000003B  03F8  page sequence number 59        |
|                              0000  03FC  page TSN base 0                |
|                              0000  03FE  MBZ '..'                       |
+-------------------------------------------------------------------------+
```

The following is an example of a snapshot data page information display:

```
+-------------------------------------------------------------------------+
|                 4001 00000001  0000  page 1, physical area 1 (snap)      |
|                      A46ACD6A  0006  checksum = A46ACD6A                 |
|              009BA304 993A8B26  000A  time stamp = 12-SEP-1997 13:02:33.60  |
|                      0000 0054  0012  84 free bytes, 0 locked            |
|                          0003  0016  3 lines                            |
|                      0000 0000  0018  line 0: empty                     |
|                      01AE 0238  001C  line 1: offset 0238, 430 bytes    |
|                      01AE 008A  0020  line 2: offset 008A, 430 bytes    |
|                       00000000  0024  line 0: TSN 0                     |
|                       000085BD  0028  line 1: TSN 34237                 |
|                       000085BF  002C  line 2: TSN 34239                 |
|                           0000  0030  line 0 -> live line: 0            |
|                           0000  0032  line 1 -> live line: 0            |
|                           0000  0034  line 2 -> live line: 0            |
|                       000001AB  03E6  live page pointer 427             |
|                       000085C4  03EA  max TSN 34244                     |
|                       FFFFFFFF  03EE  snap page pointer -1              |
|                       00000000  03F2  snap pointer TSN 0                |
|                           0000  03F6  MBZ '..'                          |
|                       00000000  03F8  page sequence number 0            |
|                           0000  03FC  page TSN base 0                   |
|                           0000  03FE  MBZ '..'                          |
+-------------------------------------------------------------------------+
```

The following is an example of an area inventory page (AIP) page information display:

```
+-------------------------------------------------------------------------+
|                 0001 000001D0  0000  page 464, physical area 1 (AIP)     |
|                      D992664E  0006  checksum = D992664E                 |
|              009646FF 8BFE44E0  000A  time stamp =  1-DEC-1992 12:49:48.  |
|                      0000 0022  0012  34 free bytes, 0 locked            |
|                       000001D1  0016  next area inventory page 465      |
|                           4001  03F6  logical area 16385                |
|                       00000000  03F8  page sequence number 0            |
|                           0000  03FC  page TSN base 0                   |
+-------------------------------------------------------------------------+
```

The following is an example of a SPAM page information display; note that a SPAM page display is quite lengthy:

```
+-------------------------------------------------------------------------+
|                    0001 00000001  0000  page 1, physical area 1 (SPAM)  |
|                         4681E156  0006  checksum = 4681E156             |
|                80000000 00000060  000A  Fast incremental backup TSN = 0:96  |
|                         0000 0001  0012  1 free byte, 0 locked          |
| FFFFFFFFFFFFFFFCFFFFFFFFFFFFFFF  0016  pages 2-31: threshold 3          |
|                                        page 32: threshold 0             |
|                                        pages 33-65: threshold 3         |
| FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF  0026  pages 66-129: threshold 3       |
| FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF  0036  pages 130-193: threshold 3      |
| FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF  0046  pages 194-257: threshold 3      |
| FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF  0056  pages 258-321: threshold 3      |
| 0FFFFFFC33C3FFFFFFFFFFFFFFFFFFF  0066  pages 322-362: threshold 3       |
|                                        pages 363-364: threshold 0       |
|                                        pages 365-366: threshold 3       |
|                                        page 367: threshold 0            |
|                                        page 368: threshold 3            |
|                                        pages 369-370: threshold 0       |
|                                        pages 371-383: threshold 3       |
|                                        pages 384-385: threshold 0       |
| 33FFFFFF03FFFFFC03FFFFF3FF0FFFFF  0076  pages 386-395: threshold 3      |
|                                        pages 396-397: threshold 0       |
|                                        pages 398-402: threshold 3       |
|                                        page 403: threshold 0            |
|                                        pages 404-414: threshold 3       |
|                                        pages 415-418: threshold 0       |
|                                        pages 419-430: threshold 3       |
|                                        pages 431-433: threshold 0       |
|                                        pages 434-446: threshold 3       |
|                                        page 447: threshold 0            |
|                                        page 448: threshold 3            |
|                                        page 449: threshold 0            |
|                                        page 513: threshold 0            |
   .
   .
   .
|                            0052  03E7  pages 1055-1057, logical area 82 |
|                            0052  03E9  pages 1058-1060, logical area 82 |
|                            0052  03EB  pages 1061-1063, logical area 82 |
|                            0052  03ED  pages 1064-1066, logical area 82 |
|                            0052  03EF  pages 1067-1069, logical area 82 |
|                            0052  03F1  pages 1070-1072, logical area 82 |
|                            0052  03F3  pages 1073-1075, logical area 82 |
|                            0052  03F5  pages 1076-1078, logical area 82 |
|                            0052  03F7  pages 1079-1081, logical area 82 |
|                            0052  03F9  pages 1082-1084, logical area 82 |
|                            0052  03FB  pages 1085-1087, logical area 82 |
|                            0052  03FD  pages 1088-1090, logical area 82 |
|                              00  03FF  MBZ free '.'                     |
+-------------------------------------------------------------------------+
```

7.3.6 RMU/SHOW STATISTIC "Logical Area" Menu Filter Option

Using the RMU/SHOW STATISTIC utility Logical Area menu was difficult when production databases contained hundreds or thousands of logical areas. One database required accessing the MORE option 230 times to get to the desired logical area.

The contents of the logical area menu can now be controlled by the use of wildcard selection criteria.

A new option has been added to the Tools menu, obtained using the exclamation mark (!) from any screen. The new Logical Area Menu Filter option lets you specify a search pattern containing wildcards.

Note

The specified pattern *MUST* match at least one logical area, or the pattern will be rejected.

The filtered logical area menu is only available when displaying all logical areas. It is *not* available if you selected the Display Application Logical Areas option from the Tools menu.

The specified pattern string may contain either one or both of the two wildcard characters, asterisk (*) and percent (%). The asterisk character is mapped to zero or more characters. The percent character is mapped to only one character. For example, the pattern "*EMP*" will find *any* logical area containing the text "EMP", while the pattern "EMP*" will find only those logical areas whose name starts with "EMP".

7.3.7 RMU/SHOW STATISTIC "Stall Messages" Screen Allows Wildcards

The RMU/SHOW STATISTIC utility Stall Messages screen Filter onscreen-menu option now allows the use of wildcards in the filtering criteria.

The pattern string may contain either one or both of the two wildcard characters, asterisk (*) and percent (%). The asterisk character is mapped to zero or more characters. The percent character is mapped to only one character.

7.3.8 CPU Time Displayed Correctly

Previously, the Oracle Rdb RMU/SHOW STATISTICS interface was unable to correctly display process CPU times in excess of 1 day; the number of days value was not displayed.

Oracle Rdb RMU/SHOW STATISTICS is now able to display CPU times greater than one day. Because the width of the CPU time display is limited, the following CPU time display formats are used:

- For CPU time values less than 1 day: "HH:MM:SS.CC"
- For CPU time values less than 100 days but more than 1 day: "DD HH:MM"
- For CPU time values more than 100 days: "DDD HH:MM"

8

Enhancements in Oracle Rdb7 Release 7.0.1.2

This chapter describes the enhancements that are introduced in Oracle Rdb7 Release 7.0.1.2.

8.1 Enhancements In All Interfaces

8.1.1 Monitor Process Uses Less ENQLM

OpenVMS platforms only.

The Oracle Rdb7 monitor process holds null mode locks on a number of database resources in order to keep the lock value blocks valid even when there are no users attached to a database. For systems that have a large number of databases or databases with a large number of storage areas, the monitor process can consume a great number of locks, sometimes exceeding its lock quota (ENQLM) even at the OpenVMS maximum value of 32767 locks.

The impact of this situation has been reduced in Oracle Rdb7 Release 7.0.1.2. By using the LCK$M_NOQUOTA flag when taking out many of these locks (in particular, the database FILID, SAC, RCACHE, TSNBLK and SEQBLK locks), the monitor process uses less of its ENQLM. The total number of locks and system resources consumed remains the same but the monitor process's ENQLM is not deducted for these locks.

8.1.2 RDMS$TTB_HASH_SIZE Logical Name

Temporary table users should be aware of a new logical name being added to Oracle Rdb7. The temporary table code uses a hash table that is sized according to the setting of the RDMS$TTB_HASH_SIZE logical name. If the logical has not been defined, a default value of 1249 will be used.

If expected usage is such that temporary tables will be large (10k or more rows), this logical should be used to adjust the hash table size used to avoid long hash chains. The setting of this logical should be on the order of roughly 1/4 of the expected maximum number of rows per temporary table. So, if its likely that a temporary table will be populated with 100,000 rows, then define this logical to be 25,000. But if there are memory constraints, it is advisable that the logical be defined no higher than this value.

8.2 SQL Interface Enhancements

8.2.1 New SQLSTATE Value

If a SQL statement expects a value from a function which does not return a value, the SQLSTATE value will be set to '2F001' to reflect the error state.

This new error code is shown in the following example.

```
SQL> CREATE DATABASE FILE TEST2;
SQL>      SET DIALECT 'SQL92';
SQL>
SQL>      CREATE MODULE RETURN_M
cont>           LANGUAGE sql
cont>
cont>           FUNCTION RETURN_F (:A INTEGER)
cont>            RETURNS INTEGER;
cont>           BEGIN
cont>           IF :A IS NOT NULL THEN
cont>         RETURN - :A;
cont>           END IF;
cont>           END;
cont>       END MODULE;
SQL>
SQL>      SELECT RETURN_F (NULL) FROM RDB$DATABASE;
%RDB-F-NORESULT, stored function "RETURN_F" returned no result
-RDB-F-ON_DB, on database SQL_USER4:[USER.DB]TEST2.RDB;
SQL> SHOW SQLCA
SQLCA:
        SQLCAID:        SQLCA           SQLCABC:        128
        SQLCODE:        -1043
        SQLERRD:        [0]: 0
                        [1]: 0
                        [2]: 0
                        [3]: 0
                        [4]: 0
                        [5]: 0
        SQLWARN0:       0       SQLWARN1:       0       SQLWARN2:       0
        SQLWARN3:       0       SQLWARN4:       0       SQLWARN5:       0
        SQLWARN6:       0       SQLWARN7:       0
        SQLSTATE:       2F001
SQL> ROLLBACK;
SQL> DROP DATABASE FILE TEST2;
```

8.2.2 Planned Change in Behavior for the UNIQUE Predicate

The next major release of Oracle Rdb will change the behavior of the UNIQUE predicate. Up to the Oracle Rdb7 release there was no semantic difference between the undocumented UNIQUE predicate and the documented SINGLE predicate. This will change with the release of Oracle Rdb8.

The UNIQUE predicate in Oracle Rdb was originally implemented for compatibility with the RDO interface and as such required that exactly one row matched, this included a single column value set to NULL. However, these semantics do not match the current SQL database language standard SQL92 for the UNIQUE predicate. Therefore, the syntax was deprecated and replaced with SINGLE.

When SINGLE is used, then a single matching row is required for uniqueness. Zero, or more than one row will be considered non-unique. The syntax and semantics of SINGLE will not be changed. If applications currently use the UNIQUE predicate, but require these semantics, then applications must be changed to use the SINGLE predicate.

The syntax for UNIQUE has been deprecated for many versions in preparation for this change in behavior in compliance with the current SQL database language standard. An example of the deprecated message, follows:

```
SQL> SELECT EMPLOYEE_ID
cont> FROM EMPLOYEES
cont> WHERE UNIQUE (SELECT EMPLOYEE_ID FROM JOB_HISTORY);
%SQL-I-DEPR_FEATURE, Deprecated Feature: UNIQUE is replaced by SINGLE
```

In Oracle Rdb8 the UNIQUE predicate will be documented and the deprecated message will no longer be used. The changed semantics may cause additional rows to be returned from queries, because now rows with column values set to NULL will always be considered UNIQUE.

Note

This topic is an announcement of a future new feature for Oracle Rdb8. Use the information contained in it for planning purposes only with Oracle Rdb7 Release 7.0.1.2.

8.2.3 UNION ALL and Derived Tables Allow up to 2000 Value Expressions

The DISTINCT, ORDER BY, GROUP BY, and UNION clauses are restricted to 255 value expressions in all releases of Rdb7 due to restrictions in processing DISTINCT and ORDER BY clauses.

Unlike UNION, the UNION ALL clause does not perform an implicit DISTINCT operation and so need not be restricted in the same way as the UNION clause. Therefore, in Oracle Rdb7 Release 7.0.1.2 the UNION ALL clause now allows up to 2000 value expressions.

The restriction of 255 column names for a derived table has also been lifted so that now up to 2000 columns can be visible through a derived table expression.

If older versions of Oracle Rdb7 are remotely accessed, then the previous limits will still be imposed.

8.3 Oracle RMU Enhancements

8.3.1 RMU/DUMP/AFTER Command /START and /END Qualifiers Improved

The /START and /END qualifiers for the RMU/DUMP/AFTER_JOURNAL command were difficult to use because users seldom know, nor can they determine, the AIJ record number in advance of using the command.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.2. The RMU/DUMP/AFTER_JOURNAL command has been enhanced to provide more advanced selection criteria. Three new optional qualifiers, /FIRST=select_list, /LAST=select_list, and /ONLY=select_list have been added.

The select_list clause of these qualifiers consists of a list of one or more of the following keywords:

- TSN=tsn

  Specifies the first, last, or specific TSN in the AIJ journal, using the standard "[n:]m" TSN format.

- TID=tid

  Specifies the first, last or specific TID in the AIJ journal.

- RECORD=record

Specifies the first or last record in the AIJ journal. This is the same as the existing /START and /END qualifiers, which are still supported, but obsolete. This keyword cannot be used with the /ONLY qualifier.

- BLOCK=block#

  Specifies the first or last block in the AIJ journal. This keyword cannot be used with the /ONLY qualifier.

- TIME=date_time

  Specifies the first or last date/time in the AIJ journal, using the standard date/time format. This keyword cannot be used with the /ONLY qualifier.

The /FIRST, /LAST, and /ONLY qualifiers are optional. You may specify any or none of them.

The keywords specified for the /FIRST qualifier can differ from the keywords specified for the other qualifiers.

For example, to start the dump from the fifth block of the AIJ journal, you would use the following command:

```
RMU/DUMP/AFTER_JOURNAL /FIRST=(BLOCK=5) MF_PERSONNEL.AIJ
```

To start the dump from block 100 or TSN 52, whichever occurs first, you would use the following command:

```
RMU/DUMP/AFTER_JOURNAL /FIRST=(BLOCK=100,TSN=0:52) MF_PERSONNEL.AIJ
```

When multiple keywords are specified for a qualifier, the first condition being encountered activates the qualifier. In the above example, the dump will start when *either* block 100 or TSN 52 is encountered.

Note

Be careful when searching for TSNs or TIDs, as they are not ordered in the AIJ journal. For example, if you want to search for a specific TSN then use the /ONLY qualifier, not the /FIRST and /LAST qualifiers.

For example, assume the AIJ journal contains records for TSN 150, 170 and 160 (in that order). If you specify the /FIRST=TSN=160 and /LAST=TSN=160 qualifiers, nothing will be dumped because the TSN 170 will match the /LAST=TSN=160 criteria.

8.3.2 RMU/SHOW STATISTICS "Stall Message Logfile" Option Real Time Lock Information

The RMU/SHOW STATISTICS utility “Stall Message Logging” facility now shows expanded information for DBAs. It now displays real-time lock information when the displayed stall is on a lock or locked object. Both the waiting process and the blocking process are displayed. The RMU/SHOW STATISTICS “Stall Message Logging” facility now provides real-time lock information when the displayed stall is on a lock or locked object.

The following example shows the new output of a sample stall messages logfile.

```
 Oracle Rdb X7.0-00 Performance Monitor Stall Log
 Database USR1$:[WORK.STATS]MF_PERSONNEL.RDB;1
 Stall Log created 30-SEP-1997 07:01:15.64
2AA8587A:1 08:11:54.27 reading pages 11:7534416 to 3:78
2AAA9E7B:1 08:11:54.31 waiting for async-write of pages 5:1412 to 5:1412
2AA810A7:1 08:11:54.29 waiting for page 5:876 (PW)
           State... Process.ID Process.name... Lock.ID. Rq Gr Queue page 876
           Blocker: 2AAA9E7B   RICK10......... 7D00562C PR PR Grant
           Waiting: 2AA810A7   RICK13......... 71002E7D PW NL Wait
2AA8587A:1 08:11:55.34 waiting for page 5:1303 (PW)
           State... Process.ID Process.name... Lock.ID. Rq Gr Queue page 1303
           Owner:   2AA7D07C   RICK11......... 31007E07 PR CR Grant
           Blocker: 2AAA9E7B   RICK10......... 5A00BA0E PR PR Grant
           Waiting: 2AA8587A   RICK9.......... 5C005FAD PW CR Cnvrt
2AAA9E7B:1 08:11:55.37 locking page 5:565
2AA810A7:1 08:11:55.38 reading pages 5:912 to 5:914
2AAA9E7B:1 08:11:57.77 waiting for page 5:1303 (PW)
           State... Process.ID Process.name... Lock.ID. Rq Gr Queue page 1303
           Owner:   2AA810A7   RICK13......... 0C007752 PR CR Grant
           Blocker: 2AA8587A   RICK9.......... 2D001C3D PR PR Grant
           Waiting: 2AAA9E7B   RICK10......... 47003DC3 PW CR Cnvrt
2AA7D07C:1 08:11:57.78 reading pages 5:1337 to 5:1339
2AA8587A:1 08:11:57.86 reading pages 5:330 to 5:332
2AA7D07C:1 08:11:57.86 waiting for page 5:1413 (PR)
           State... Process.ID Process.name... Lock.ID. Rq Gr Queue page 1413
           Blocker: 2AAA9E7B   RICK10......... 6A002CBB PW PW Grant
           Owner:   2AA8587A   RICK9.......... 6F008623 PR CR Grant
           Waiting: 2AA7D07C   RICK11......... 1F007B4D PR NL Wait
   .
   .
   .
```

8.3.3 RMU/SHOW STATISTICS Utility "Stall Messages Log" Displays Stall Duration Information

The RMU/SHOW STATISTICS utility “Stall Messages Logging” facility has been enhanced to provide the information necessary to determine stall duration.

First, the current time has been added to each stall message. This allows you to determine the stall duration at that point-in-time, because the stall start-time is also displayed.

Secondly, a new qualifier has been added: /OPTION=VERBOSE. This qualifier causes the stall message logging facility to report a stall message at each interval, even if it has been previously reported.

Note

Use of the /OPTION=VERBOSE qualifier could result in an enormous stall messages logfile. Ensure that adequate disk space exists for the logfile when using this qualifier.

The stall messages logging “verbose” option can be enabled and disabled at runtime, using the “Tools” menu, by pressing the “!” key.

The verbose option can also be specified in the configuration file, using the STALL_LOG_VERBOSE variable. Valid keywords are ENABLED or DISABLED.

The following example shows a stall messages logfile, in "verbose" mode, for a database where four processes are all stalled on the same lock. Note that the first stall message already indicates a 25-minute stall.

```
 Oracle Rdb X7.0-00 Performance Monitor Stall Log
 Database USR1$:[WORK.STATS]MF_PERSONNEL.RDB;1
 Stall Log created  2-OCT-1997 09:26:15.19
09:26:15.19 2AA8C6D7:1 09:01:01.29 waiting for logical area 58 (CW)
            State... Process.ID Process.name... Lock.ID. Rq Gr Queue logical
area 58
            Blocker: 2AA00443   RICK2..... 7300845F PW PW Grant
            Waiting: 2AA8C6D7   RICK6..... 4E008184 CW NL Cnvrt
            Waiting: 2AA912D8   RICK7..... 5D0034F2 CW NL Cnvrt
            Waiting: 2AA3BADC   RICK8..... 0700115F CW NL Cnvrt
            Waiting: 2AA43ADE   RICK9..... 4700AE41 CW NL Cnvrt
09:26:15.19 2AA3BADC:1 09:01:01.37 waiting for logical area 58 (CW)
            State... Process.ID Process.name... Lock.ID. Rq Gr Queue logical
area 58
            Blocker: 2AA00443   RICK2..... 7300845F PW PW Grant
            Waiting: 2AA8C6D7   RICK6..... 4E008184 CW NL Cnvrt
            Waiting: 2AA912D8   RICK7..... 5D0034F2 CW NL Cnvrt
            Waiting: 2AA3BADC   RICK8..... 0700115F CW NL Cnvrt
            Waiting: 2AA43ADE   RICK9..... 4700AE41 CW NL Cnvrt
09:26:15.19 2AA912D8:1 09:01:01.32 waiting for logical area 58 (CW)
            State... Process.ID Process.name... Lock.ID. Rq Gr Queue logical
area 58
            Blocker: 2AA00443   RICK2..... 7300845F PW PW Grant
            Waiting: 2AA8C6D7   RICK6..... 4E008184 CW NL Cnvrt
            Waiting: 2AA912D8   RICK7..... 5D0034F2 CW NL Cnvrt
            Waiting: 2AA3BADC   RICK8..... 0700115F CW NL Cnvrt
            Waiting: 2AA43ADE   RICK9..... 4700AE41 CW NL Cnvrt
   .
   .
   .
```

The lock information is only displayed once per stall, even in verbose mode, to minimize the the output file size.

8.3.4 RMU/SHOW STATISTICS "User-Defined Events" Enhancements

The following enhancements have been made to the RMU/SHOW STATISTICS utility "User-Defined Events" facility and the "Configuration File" facility in general:

- Long configuration file lines can be continued on the next line by terminating the line with a back-slash ("\"). Lines can be continued up to 2048 characters, even within quoted values; for example:

  ```
  EVENT_DESCRIPTION="ENABLE 'pages checked' MAX_CUR_TOTAL \
   INITIAL 7 \
   EVERY 11 \
   LIMIT 100 \
   INVOKE DB_ALERT";
  ```

  This enhancement is not limited to just the EVENT_DESCRIPTION variable; it can be used for any configuration variable. Also, comments can be embedded in continued lines if they start at the beginning of the next line. For example, consider the following two event descriptions:

```
EVENT_DESCRIPTION="ENABLE '  (Asynch. reads)' MAX_CUR_TOTAL \
! this will work as expected
 AREA EMPIDS_OVER \
 INITIAL 6 EVERY 10 LIMIT 100 INVOKE DB_ALERT";
EVENT_DESCRIPTION="ENABLE '  (Asynch. reads)' MAX_CUR_TOTAL \
 AREA EMPIDS_OVER ! this will NOT work as expected \
 INITIAL 6 EVERY 10 LIMIT 100 INVOKE DB_ALERT";
```

Note that the comment in the second event description takes precedence over the line continuation character.

- In the EVENT_DESCRIPTION variable value, the underscore character ("_") or dash character ("-") can be used in place of spaces in statistics names that have leading spaces. For example, the statistics field name " file extend" can also be specified as "_ _ file_extend" or "- - file-extend". This is useful for improving the readability of difficult statistics field names.
- The keyword "AREA" has been added to the user-defined event attribute list. The keyword "AREA" allows you to specify the name of a storage area. When this keyword is specified, the statistics field selected must be from the "IO Statistics (by file)" or "Locking Statistics (by file)" screens.

  The **AREA** attribute is available when using the **/NOINTERACTIVE** qualifier, or when using the "INTERACTIVE" configuration variable set to **FALSE**.
- The keyword "LAREA" has been added to the user-defined event attribute list. The keyword "LAREA" allows you to specify the name of a logical area, which can be either a table, B-tree index, hash index or blob. When this keyword is specified, the statistics field selected must be from the "Logical Area" screens.

  If the logical area is partitioned across multiple storage areas, the keyword "AREA" can be used to identify a specific partition to define the event against.

  The LAREA attribute is available when using the /NOINTERACTIVE qualifier, or when using the "INTERACTIVE" configuration variable set to FALSE.

The following table explains the semantics of specifying the AREA and LAREA keywords:

AREA	LAREA	Description
No	No	Regular statistic field used
Yes	No	Storage Area statistic field used
No	Yes	Logical Area statistic field used - all partitions
Yes	Yes	Logical Area statistic field used - single partition

This example demonstrates how to define an event on a storage area statistic:

```
EVENT_DESCRIPTION="ENABLE '  (Asynch. reads)' MAX_CUR_TOTAL \
 AREA EMPIDS_OVER \
 INITIAL 6 EVERY 10 LIMIT 100 INVOKE DB_ALERT";
```

This example demonstrates how to define an event on a table. Note that this event is defined across all partitions of the table.

```
EVENT_DESCRIPTION="ENABLE 'pages checked' MAX_CUR_TOTAL \
 LAREA EMPLOYEES \
 INITIAL 1 EVERY 1 LIMIT 100 INVOKE DB_ALERT";
```

This example demonstrates how to define an event on a single-partition of a partitioned table.

```
EVENT_DESCRIPTION="ENABLE 'pages checked' MAX_CUR_TOTAL \
 LAREA EMPLOYEES AREA EMPIDS_LOW \
 INITIAL 3 EVERY 7 LIMIT 100 INVOKE DB_ALERT";
```

The "Statistics Event Information" screen has been enhanced to identify the physical area ID and logical area ID for each event. The area identifiers are displayed when using "Full" display-mode. For example, using the above examples, the screen would appear as follows:

```
Node: ALPH   (1/1/2)    Oracle Rdb X7.0-00 Perf. Monitor 21-OCT-1997 13:41:50.06
Rate: 1.00 Second         Statistics Event Information      Elapsed: 02:30:21.57
Page: 1 of 1             DISK$:[WORK]MF_PERSONNEL.RDB;1     Mode: Online
-------------------------------------------------------------------------------
Statistic..... Event..... State...     Threshold Every       Current  Cnt
Program/Operator.Notification..... Parea Larea Rem Limit
synch data reads   MAX_CUR_TOTAL enabled          228.0    11         228.0    1
DB_ALERT (@SYS$DISK:[]EVENT.COM)                              0     0   0   100
locks requested    MAX_CUR_TOTAL enabled          406.0    10         406.0    1
DB_ALERT (@SYS$DISK:[]EVENT.COM)                              5     0   0   100
pages checked      MAX_CUR_TOTAL enabled            3.0     7           0.0    0
DB_ALERT (@SYS$DISK:[]EVENT.COM)                              3    56   0   100
pages checked      MAX_CUR_TOTAL enabled            4.0     8           0.0    0
DB_ALERT (@SYS$DISK:[]EVENT.COM)                              4    57   0   100
pages checked      MAX_CUR_TOTAL enabled        10717.0     9       10717.0    1
DB_ALERT (@SYS$DISK:[]EVENT.COM)                              5    58   0   100
pages checked      MAX_CUR_TOTAL enabled        10717.0     1       10717.0    1
DB_ALERT (@SYS$DISK:[]EVENT.COM)                              0     2   0   100

-------------------------------------------------------------------------------
Brief Config Exit Help Menu >next_page <prev_page Options Pause Set_rate Write
```

If an event on a storage area or logical area is encountered, the storage area name, the logical area name, or both are passed as the eighth parameter to the invocation program. For example, the DB_INVOKE program defined above causes the DCL script "EVENT.COM" to be executed. This DCL script simply appends the raised event to a log file; for example:

```
$ SET NOON
$ OPEN/APPEND/SHARE=READ EVENT_LOG SYS$DISK:[]EVENT.LOG
$ WRITE EVENT_LOG "''P1' ''P2' ''P3' ''P4' ''P5' ''P6' COUNT IS ''P7' ''P8'"
$ CLOSE EVENT_LOG
$ EXIT
```

Note that the "P8" parameter is either null ("") or contains the name of the target storage area or logical area. The following is an example of the logfile output:

```
20-OCT-1997 14:02:21.41 pages checked MAX_CUR_TOTAL 6.0 above 4.0 count is 1
EMPIDS_MID.EMPLOYEES
20-OCT-1997 14:02:22.16 pages checked MAX_CUR_TOTAL 32820.0 above 5.0 count is 1
EMPIDS_OVER.EMPLOYEES
```

Note that when both the storage area and logical area names are specified, they are separated by a period (".").

8.3.5 Added Detail to RMU/SHOW STATISTICS "SPAM Fetches" Screen

The RMU/SHOW STATISTICS utility "SPAM Fetches" screen did not display the reason why a SPAM page was fetched. This information is vital in determining when excessive SPAM fetches are occurring.

The following example shows a sample "PIO Statistics–SPAM Fetches" screen display. It is extremely difficult to determine what caused the 17,821 SPAM fetches as well as the 2,250 SPAM updates.

```
Node: ALPH   (1/1/16)   Oracle Rdb X7.0-00 Perf. Monitor  5-JAN-1998 11:27:51.18
Rate: 1.00 Second         PIO Statistics--SPAM Fetches      Elapsed: 00:30:17.60
Page: 1 of 1            DISK1$:[WORK]MF_PERSONNEL.RDB;1      Mode: Online
--------------------------------------------------------------------------------

statistic.........      rate.per.second.............. total....... average......
name..............      max..... cur..... avg....... count....... per.trans....

fetch for read                20        0        9.8        17821           1.0
fetch for write               47        0        1.2         2250           0.1

in LB: all ok                 60        0       11.0        20031           1.2
   LB: need lock               1        0        0.0           39           0.0
   LB: old version             0        0        0.0            0           0.0

not found: read                0        0        0.0            1           0.0
         : synth               0        0        0.0            0           0.0

DAPF: success                  0        0        0.0            0           0.0
DAPF: failure                  0        0        0.0            0           0.0
DAPF: utilized                 0        0        0.0            0           0.0
DAPF: discarded                0        0        0.0            0           0.0

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

This problem has been corrected in Oracle Rdb7 Release 7.0.1.2. The RMU/SHOW STATISTICS utility has been enhanced with the "PIO Statistics–SPAM Access" screen. The purpose of this screen is to identify the reason why the SPAM was accessed, for either read or write.

Using the same database as the above example, consider the following screen:

```
Node: ALPH   (1/1/16)   Oracle Rdb X7.0-00 Perf. Monitor  5-JAN-1998 11:29:21.13
Rate: 1.00 Second         PIO Statistics--SPAM Access       Elapsed: 00:31:47.55
Page: 1 of 1            DISK$:[WORK]MF_PERSONNEL.RDB;1      Mode: Online
--------------------------------------------------------------------------------

statistic.........      rate.per.second.............. total....... average......
name..............      max..... cur..... avg....... count....... per.trans....
fetch for read               20        0        9.3        17821           1.0
 uniform area scan           15        0        0.1          280           0.0
 record store fet            20        0        9.1        17541           1.0
 record modify fet            0        0        0.0            0           0.0
 record erase fet             0        0        0.0            0           0.0
fetch for write              47        0        1.1         2250           0.1
 record store upd             4        0        0.4          858           0.0
 record modify upd            0        0        0.0            0           0.0
 record erase upd            23        0        0.1          321           0.0
fetch for update             47        0        1.1         2250           0.1
 clump allocate               3        0        0.1          216           0.0
 fast incr. bkup              0        0        0.0            0           0.0
 threshold update            23        0        0.5          963           0.0
record stored                16        0        8.7        16677           1.0
record marked              1849        0       22.0        42049           2.5
record erased               622        0        4.3         8338           0.5

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

As can be clearly seen, the majority of the SPAM page "fetch for read" accesses were caused by record storage. The SPAM page "fetch for write" accesses are evenly distributed between record stores and record erases.

Note that 16,677 records were stored, while 17,541 SPAM fetches occurred because of those stores. However, only 858 of those SPAM fetches actually resulted in updates to the SPAM thresholds.

Excessive SPAM fetches can be identified by comparing the "record store fet" field against the "record store upd" and "record stored" fields.

Table 8–1 "SPAM Access" Screen Fields

Field	Description
fetch for read	The total number of times the SPAM page was fetched for retrieval.
uniform area scan	The total number of times the SPAM page was fetched for retrieval during a record store operation. This is used primarily to check if SPAM thresholds need to be adjusted.
record store fet	The total number of times the SPAM page was fetched for retrieval during a record store operation. This is primarily used to check if SPAM thresholds need to be adjusted.
record modify fet	The total number of times the SPAM page was fetched for retrieval during a record modification operation. This is primarily used to check if SPAM thresholds need to be adjusted.
record erase fet	The total number of times the SPAM page was fetched for retrieval during a record erase operation. This is primarily used to check if SPAM thresholds need to be adjusted.

(continued on next page)

Table 8–1 (Cont.) "SPAM Access" Screen Fields

Field	Description
fetch for write	The total number of times the SPAM page was fetched for update.
record store upd	The total number of times the SPAM page was fetched for update during a record store operation. This is primarily used to modify the SPAM thresholds.
record modify upd	The total number of times the SPAM page was fetched for update during a record modification operation. This is primarily used to modify the SPAM thresholds.
record erase upd	The total number of times the SPAM page was fetched for update during a record erase operation. This is primarily used to modify the SPAM thresholds.
fetch for update	The total number of times the SPAM page was fetched for update.
clump allocate	The total number of times the SPAM page was updated for a clump allocation operation.
fast incr. bkup	The total number of times the SPAM page was updated for a fast incremental backup modification.
threshold update	The total number of times the SPAM page was updated to change a data page's threshold information.
record stored	The total number of records stored.
record marked	The total number of records modified.
record erased	The total number of records erased.

The "PIO Statistics–SPAM Access" screen is recorded to the binary output file, and is available during binary input file replay.

The following example shows the statistics collected following an operation that stored 8,192 records into an uniform-format storage area:

```
Node: ALPH   (1/1/16)   Oracle Rdb X7.0-00 Perf. Monitor  5-JAN-1998 12:20:06.57
Rate: 1.00 Second          PIO Statistics--SPAM Access       Elapsed: 00:10:42.88
Page: 1 of 1             DISK$:[WORK]MF_PERSONNEL.RDB;1      Mode: Online
------------------------------------------------------------------------------

statistic.........      rate.per.second.............. total....... average......
name..............      max..... cur..... avg....... count....... per.trans....
fetch for read               19        0       13.6         8775           1.0
 uniform area scan            1        0        0.1          110           0.0
 record store fet            17        0       13.4         8665           1.0
 record modify fet            0        0        0.0            0           0.0
 record erase fet             0        0        0.0            0           0.0
fetch for write               3        0        0.9          642           0.0
 record store upd             1        0        0.4          321           0.0
 record modify upd            0        0        0.0            0           0.0
 record erase upd             0        0        0.0            0           0.0
fetch for update              3        0        0.9          642           0.0
 clump allocate               0        0        0.0            0           0.0
 fast incr. bkup              0        0        0.0            0           0.0
 threshold update             1        0        0.4          321           0.0
record stored                16        0       12.9         8338           1.0
record marked                17        0       13.4         8661           1.0
record erased                 0        0        0.0            0           0.0

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

The following example shows the statistics collected following an operation that scanned 8,192 records into an uniform format storage area:

```
Node: ALPH   (1/1/16)   Oracle Rdb X7.0-00 Perf. Monitor  5-JAN-1998 12:42:44.65
Rate: 1.00 Second         PIO Statistics--SPAM Access       Elapsed: 00:01:25.56
Page: 1 of 1            DISK$:[WORK]MF_PERSONNEL.RDB;1      Mode: Online
------------------------------------------------------------------------------

statistic.........      rate.per.second.............. total....... average......
name..............      max..... cur..... avg....... count....... per.trans....
fetch for read               158        0        3.8          330         165.0
 uniform area scan           158        0        3.8          330         165.0
 record store fet              0        0        0.0            0           0.0
 record modify fet             0        0        0.0            0           0.0
 record erase fet              0        0        0.0            0           0.0
fetch for write                0        0        0.0            0           0.0
 record store upd              0        0        0.0            0           0.0
 record modify upd             0        0        0.0            0           0.0
 record erase upd              0        0        0.0            0           0.0
fetch for update               0        0        0.0            0           0.0
 clump allocate                0        0        0.0            0           0.0
 fast incr. bkup               0        0        0.0            0           0.0
 threshold update              0        0        0.0            0           0.0
record stored                  0        0        0.0            0           0.0
record marked                  0        0        0.0            0           0.0
record erased                  0        0        0.0            0           0.0

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

The following example shows the statistics collected following an operation that modified 8,192 records into an uniform format storage area:

```
Node: ALPH   (1/1/16)   Oracle Rdb X7.0-00 Perf. Monitor  5-JAN-1998 12:24:44.15
Rate: 1.00 Second         PIO Statistics--SPAM Access       Elapsed: 00:03:34.91
Page: 1 of 1            DISK$:[WORK]MF_PERSONNEL.RDB;1      Mode: Online
------------------------------------------------------------------------------

statistic.........      rate.per.second.............. total....... average......
name..............      max..... cur..... avg....... count....... per.trans....
fetch for read               543        0       31.4         6765        3382.5
 uniform area scan           147        0        1.9          415         207.5
 record store fet            529        0       29.5         6350        3175.0
 record modify fet             0        0        0.0            0           0.0
 record erase fet              0        0        0.0            0           0.0
fetch for write               63        0        3.3          711         355.5
 record store upd             34        0        1.8          395         197.5
 record modify upd             0        0        0.0            0           0.0
 record erase upd              0        0        0.0            0           0.0
fetch for update              63        0        3.3          711         355.5
 clump allocate               14        0        0.7          158          79.0
 fast incr. bkup               0        0        0.0            0           0.0
 threshold update             20        0        1.1          237         118.5
record stored                494        0       26.5         5703        2851.5
record marked                703        0       38.1         8192        4096.0
record erased                  0        0        0.0            0           0.0

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

The following example shows the number of SPAM pages retrieved in order to store a single, new record:

```
Node: ALPH   (1/1/16)   Oracle Rdb X7.0-00 Perf. Monitor  5-JAN-1998 12:25:55.48
Rate: 1.00 Second         PIO Statistics--SPAM Access       Elapsed: 00:00:24.01
Page: 1 of 1            DISK$:[WORK]MF_PERSONNEL.RDB;1      Mode: Online
-------------------------------------------------------------------------------

statistic.........      rate.per.second.............. total....... average......
name..............      max..... cur..... avg....... count....... per.trans....
fetch for read               720        0       31.4          756         756.0
 uniform area scan           180        0        7.9          190         190.0
 record store fet            539        0       23.5          566         566.0
 record modify fet             0        0        0.0            0           0.0
 record erase fet              0        0        0.0            0           0.0
fetch for write                1        0        0.0            2           2.0
 record store upd              0        0        0.0            1           1.0
 record modify upd             0        0        0.0            0           0.0
 record erase upd              0        0        0.0            0           0.0
fetch for update               1        0        0.0            2           2.0
 clump allocate                0        0        0.0            0           0.0
 fast incr. bkup               0        0        0.0            0           0.0
 threshold update              0        0        0.0            1           1.0
record stored                  1        0        0.0            2           2.0
record marked                  1        0        0.0            2           2.0
record erased                  0        0        0.0            0           0.0

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

Now that the clump has been allocated, subsequent record storage into the same clump is significantly easier:

```
Node: ALPH   (1/1/16)   Oracle Rdb X7.0-00 Perf. Monitor  5-JAN-1998 12:27:05.47
Rate: 1.00 Second         PIO Statistics--SPAM Access       Elapsed: 00:00:36.53
Page: 1 of 1            DISK$:[WORK]MF_PERSONNEL.RDB;1      Mode: Online
-------------------------------------------------------------------------------

statistic.........      rate.per.second.............. total....... average......
name..............      max..... cur..... avg....... count....... per.trans....
fetch for read               621        0       17.8          653         326.5
 uniform area scan           156        0        4.4          164          82.0
 record store fet            465        0       13.3          489         244.5
 record modify fet             0        0        0.0            0           0.0
 record erase fet              0        0        0.0            0           0.0
fetch for write                0        0        0.0            0           0.0
 record store upd              0        0        0.0            0           0.0
 record modify upd             0        0        0.0            0           0.0
 record erase upd              0        0        0.0            0           0.0
fetch for update               0        0        0.0            0           0.0
 clump allocate                0        0        0.0            0           0.0
 fast incr. bkup               0        0        0.0            0           0.0
 threshold update              0        0        0.0            0           0.0
record stored                  0        0        0.0            1           0.5
record marked                  0        0        0.0            1           0.5
record erased                  0        0        0.0            0           0.0

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

8.3.6 RMU/SHOW STATISTICS Enhanced to Prevent Database Hangs

It was sometimes possible for the RMU/SHOW STATISTIC utility to cause the database to hang. This could occur when the utility was left idle at a user prompt or menu request.

The database would typically hang when opening or closing the database on a different node.

The following scenario shows the problem:

After rebooting a machine the operators tried to open all the databases and one of them wouldn't open. The RMU/OPEN MYDB command did not respond.

Typing the RMU/SHOW USERS MYDB command on the same node showed the following:

```
database MYDB.RDB;1
* database startup is in progress
* database is opened by an operator
* operator is waiting for reply to open request
```

Editing the monitor log for this node showed the equivalent of:

```
   .
   .
   .
6-JAN-1998 11:08:00.01 - received open database request
from 202011C9:0
- process name _NTY150:, user JVAITKUN
- for database "MYDB" [_$1$DUA7] (34,75,0)
- database global section name is "RDM61T_3H30OND"
- database global section size is 172 pages (512 bytes per page)
- database startup waiting for MEMBIT lock
```

The database was open on all other nodes in the cluster but could not be connected to from the local node and the RMU/SHOW STATISTICS MYDB command would not work. Also, all attached processes appeared to be hung.

The problem was traced to a previously running RMU/SHOW STATISTICS screen that the operators had started. Operators were instructed to monitor the stall messages as well as to enable the alarm by pressing 'A'. In this instance they pressed 'S' for Set Rate and received the prompt "Enter time interval in seconds:" to which no one replied. As soon as a time was entered, the database opened.

The RMU/SHOW STATISTICS utility has been enhanced with the new command-line qualifier /PROMPT_TIMEOUT. This qualifier allows you to specify the user prompt timeout interval, in seconds. The default value is 60 seconds.

If you specify the /NOPROMPT_TIMEOUT qualifier or the value "0", the RMU/SHOW STATISTICS utility will not timeout any user prompts. Note that this is the current behavior and can potentially cause a database hang situation.

Note

Oracle recommends that you do not use the /NOPROMPT_TIMEOUT qualifier or the value "0" unless you are certain prompts will always be responded to in a timely manner.

If the /PROMPT_TIMEOUT qualifier is specified with a value less than ten seconds, the value "10" will be used.

The user prompt timeout interval can also be specified using the PROMPT_ TIMEOUT configuration variable.

8.3.7 New SHOW STATISTICS Utility "AIJ Backup Activity" Screen

The RMU/SHOW STATISTICS utility has been enhanced with the new "AIJ Backup Activity" screen. Located in the "Process Information" sub-menu, the "AIJ Backup Activity" screen displays information about each AIJ backup operation being performed on the node.

The "AIJ Backup Activity" screen is also available during cluster-wide statistic collection. This means you can monitor the activities of all AIJ backup operations occurring on any node accessing the database.

The "AIJ Backup Activity" screen information is not recorded in the binary output file. Therefore, the screen is not available during binary file replay.

The following example shows a sample "AIJ Backup Activity" screen:

```
Node: ALPH   (1/1/16)   Oracle Rdb X7.0-00 Perf. Monitor 19-DEC-1997 15:50:24.49
Rate: 0.50 Seconds            AIJ Backup Activity           Elapsed: 00:03:57.99
Page: 1 of 1             DISK$:[WORK]MF_PERSONNEL.RDB;1     Mode: Online
------------------------------------------------------------------------------
Process.ID  Activity... VBN...... Operation.......................... Lock.ID.
34218467:1s block bkup  7:1017    Initializing AIJ journal

------------------------------------------------------------------------------
```

The following example shows the same AIJ backup operation in a later stage of the backup operation:

```
Node: ALPH   (1/1/16)   Oracle Rdb X7.0-00 Perf. Monitor 19-DEC-1997 15:50:24.99
Rate: 0.50 Seconds            AIJ Backup Activity           Elapsed: 00:03:58.49
Page: 1 of 1             DISK$:[WORK]MF_PERSONNEL.RDB;1     Mode: Online
------------------------------------------------------------------------------
Process.ID  Activity... VBN...... Operation.......................... Lock.ID.
34218467:1s finish      7:1017    writing ROOT file (AIJFB VBN 1228)

------------------------------------------------------------------------------
```

The "AIJ Backup Activity" screen contains five columns of information:

- Process.ID: This field contains the process identifier of the AIJ backup process. This process may be the AIJ backup server (ABS), in which case the process identifier will contain the suffix "s". This process may also be the manual RMU/BACKUP/AFTER_JOURNAL utility, in which case the process identifier will contain the "u" suffix.

  Additional information can be obtained about this process by using the "Zoom" onscreen-menu option.

- Activity: This field contains a description of the backup activity being performed by the AIJ backup utility. The following backup activities are displayed:

  - activation: The AIJ backup utility is being invoked by the monitor if it is the ABS, or startup if the manual backup utility is being used.
  - bind: The AIJ backup utility is binding to the database.
  - start: The AIJ backup utility is starting the backup operation.
  - create bkup: The AIJ backup utility is creating a disk-based backup file.
  - create temp: The AIJ backup utility is creating a temporary AIJ journal. This activity typically occurs when the fast commit feature is used in conjunction with extensible AIJ journals.

  - record bkup: The AIJ backup utility is backing up an extensible AIJ journal to disk, or any type of AIJ journal to tape, using a record-by-record transfer algorithm.
  - block bkup: The AIJ backup utility is backing up a fixed-size (circular) AIJ journal to disk, using a 127-block transfer algorithm.
  - finish: The AIJ backup utility is completing the backup of an AIJ journal.
  - quiet-point: The AIJ backup utility is attempting to acquire the quiet-point lock.
  - record shfl: The AIJ backup utility is performing the record shuffle operation used for extensible AIJ journals.
  - unbind: The AIJ backup utility is unbinding from the database.

- VBN: This column identifies the current block number of the AIJ journal being backed up. The block number is normally prefixed with the AIJ sequence number, so it is easy to identify which AIJ journal is being backed up.
- Operation: This column identifies the activity-specific operation being performed by the AIJ backup utility. This column contains messages similar to those displayed by the "Stall Messages" screen.
- Lock.ID: This column identifies any lock the AIJ backup utility may be trying to acquire. This lock is typically the quiet-point lock.

  More information about this lock can be obtained using the "LockID" onscreen-menu option.

9

Enhancements in Oracle Rdb7 Release 7.0.1.3

This chapter describes the enhancements that were introduced in Oracle Rdb7 Release 7.0.1.3.

9.1 Enhancements In All Interfaces

9.1.1 Exceeding Complex Query Limit Generated %RDMS-F-MAX_CCTX Error

Prior to Oracle Rdb Release 6.0, you could generate a complex query that exceeded the limit of 32 contexts. However, when more than 32 contexts were encountered for a single query, Oracle Rdb generated the following error:

```
%RDMS-F-MAX_CCTX exceeded maximum allowable context number
```

Examples of objects in a query that would count as a context are table references, views, inner selects, or aggregates.

For Oracle Rdb Release 6.0 and later releases, the context limit was raised from 32 contexts to 128 contexts.

9.1.2 New Maximum Equivalent Class Limit for Complex Queries

Bugs 611733 and 610614.

When a query uses many nested subqueries with equality predicates, the optimizer could reach its limit of equivalent classes, At that point, the query becomes very unpredictable, and finally runs out of memory.

Oracle Rdb7 optimizer has been enhanced to increase the maximum number of equivalent classes to 1024.

9.1.3 Monitor Consumes Less Virtual Memory when Opening Databases with Global Buffers

All OpenVMS platforms.

On large systems with very large numbers of global buffers, the Oracle Rdb7 monitor process (RDMMON) could have all of its process virtual address space consumed by a very small number of databases due to the amount of virtual address space needed to map the database global section. This could prevent additional databases from being opened on the node.

In order to help relieve this virtual memory limitation of the monitor process, the global buffers portion of the database global section is no longer mapped by the monitor. This global buffers portion of the database global section is generally the largest single portion of the global section and not mapping it can greatly reduce the amount of the monitor's virtual memory consumed by the database global section. For some databases, the amount of virtual memory that the monitor requires can be a small fraction of the total database global section size.

For example, a database with 20,000 global buffers and a buffer size of 6 blocks requires 120,000 pages (Alpha pagelets) of virtual address space for the global buffers themselves. The size of the entire database global section as shown by RMU/DUMP/HEADER is 70062212 bytes (136,840 pages):

```
$ RMU/DUMP/HEADER DKA0:[DB]MYDB.RDB;1
    .
    .
    .
  Derived Data...
    - Global section size
        With global buffers disabled is 379982 bytes
        With global buffers enabled is 70062212 bytes
```

Because the global buffers are not mapped, the monitor process only maps 16,893 of the 136,840 pages for a savings of 120,000 pages of virtual memory. This savings can allow the monitor to keep more databases concurrently open before its process virtual address space would be consumed.

The following example shows a portion of the monitor log file for a database open request for a database with 11,000 global buffers. The size of the database global section is 75,631 pages but the monitor process maps only 9,631 into virtual memory.

```
16-Mar-1998 02:56:18.92 - received open database request from 22E00479:0
  - process name random@TNA23, user RDBNT
  - for database "_$1$DIA0:[DB]MYDB.RDB;1" [_$1$DIA0] (271,893,0)
  - number of global buffers is 11000, maximum buffers per user is 5
  - database global section name is "RDM70T_8K1ADR00"
  - database global section size is 75631 pages (512 bytes per page)
    - monitor maps 9631 pages of the global section into virtual memory
```

User processes continue to map the entire database global section, it is only the monitor process that does not map the global buffers portion of the global section.

9.1.4 Restrictions Lifted for Strict Partitioning

Bug 548039.

When a table's storage map has the attribute PARTITIONING IS NOT UPDATABLE, mapping of data to a storage area is strictly enforced. This is known as **strict partitioning**. This release of Oracle Rdb7 lifts restrictions imposed by earlier releases as described below.

- Strict partitioning now enforced at runtime.

  In prior releases of Oracle Rdb7, the PARTITIONING IS NOT UPDATABLE rule was enforced during query compilation. Therefore, any UPDATE statement, procedure, or trigger definition which attempted to update the partitioning columns were rejected. This created a problem for 4GL tools and generated applications which didn't know that these columns were not allowed to appear in an UPDATE statement.

  This enforcement has been lessened for this release. The enforcement is now data-value based and allows updates to these columns if the data values do not change. The prior data values are compared with the new row/column values and any changes are reported as runtime errors. If no rows are updated or the column values do not change, then the update is permitted. This allows 4GL tools and generated applications to reference these columns in a generalized UPDATE statement.

Note

A small amount of CPU time overhead is added to the UPDATE statement which must save and compare the partitioning column values. If an UPDATE statement avoids referencing these columns then this overhead is eliminated.

- Locking behavior for ISOLATION SERIALIZABLE

  In prior releases of Oracle Rdb7 when the current transaction is started using the ISOLATION SERIALIZABLE level (the default), all partitions of a table are locked in protected mode. This was done to enforce the serializable characteristic of the transaction.

  However, if a strictly partitioned query is being performed, not all the partitions need to be locked so strongly. The serializable characteristic of the transaction can be guaranteed by only locking the partitions used by the query.

  In this release of Oracle Rdb7, each partition is locked when it is referenced, and therefore concurrent sequential access to a strictly partitioned table is now possible. If the application needs to have partitions locked immediately rather than as they are referenced, or in a stronger exclusive mode, then the SET TRANSACTION .. RESERVING PARTITION clause should be used (see Section 9.2.5).

9.1.5 Date Subtraction

Some Oracle applications rely on being able to subtract one date from another and getting back the number of days between the two dates. In an effort to better support those applications, that support has been provided in the Oracle Level1 dialect.

Unlike Oracle, however, partial days are not returned. The result is always an integer value.

The following example shows the subtraction of dates:

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

          15
1 row selected
SQL>
```

9.1.6 Default Node Size Now Displayed After Index Is Created

In prior releases of Oracle Rdb7, a CREATE INDEX statement would supply a default index node size if none were provided for a UNIQUE SORTED index, or a SORTED RANKED index. However, neither the SQL SHOW INDEX, SHOW TABLE nor RMU/EXTRACT statements would display the value of this default node size.

This problem has been corrected in Oracle Rdb7 Version 7.0.1.3. All new indexes will have stored the default node size for display by SQL and RMU/EXTRACT statements.

The following example the default node size is displayed after an index is created.

```
SQL> -- Create a simple table upon which we can define
SQL> -- some indices
SQL>
SQL> CREATE TABLE TEST_INDEX_TABLE
cont>    (A CHAR(70),
cont>     B INTEGER);
SQL>
SQL> -- Default value is 430 bytes
SQL>
SQL> CREATE UNIQUE INDEX TEST_INDEX_DEF
cont>    ON TEST_INDEX_TABLE (A, B)
cont>    TYPE IS SORTED
cont>    USAGE UPDATE;
SQL>
SQL> SHOW TABLE (INDEX) TEST_INDEX_TABLE
Information for table TEST_INDEX_TABLE

TEST_INDEX_DEF                  with column A
                                and column B
  No Duplicates allowed
  Type is Sorted
  Compression is DISABLED
  Node size  430
  Percent fill  70
```

9.1.7 RUJ Buffers in a Global Section When Row Cache is Enabled

All OpenVMS platforms.

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

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

When the global section option is chosen, the RUJ information is made available to any possible future database recovery process from the shared memory global section. Traditionally, such information was only shared by writing the information to the RUJ file which the DBR process could read. By adding this capability, only an in-memory I/O is required before modifying a row in the row cache.

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

To enable this optimization, define the logical name RDM$BIND_RUJ_GLOBAL_SECTION_ENABLED to "1" in the system logical name table. The global section created for the RUJ buffers will be about 256 VAX pages or 16 Alpha Pages for each allowed user of a database. One global section will be created for each database that has row cache enabled. Databases that do not have row cache enabled will not have the RUJ global buffer optimization enabled.

The following OpenVMS system parameters will also need to be modified:

- GBLSECTIONS will need to be increased by the maximum number of Oracle Rdb databases open at one time on the system.
- GBLPAGES will need to be increased by 256 times the maximum number of users for each databases open at one time on the system.
- GBLPAGFIL will need to be increased by either 256 (on OpenVMS VAX) or 16 (on OpenVMS Alpha) times the maximum number of users for each databases open at one time on the system.

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

9.1.8 Enhancements to Range Queries on SORTED Indexes

Bug 500856.

In previous versions of Oracle Rdb, the last index key fetched from the index partition during a range query was used to determine if the scan was complete for the current range or if the next partition needed to be scanned. This could result in unnecessary scans of subsequent index partitions if the last fetched value in the SORTED index partition was not beyond the query range.

There are two important benefits to this enhancement. First, there is a reduction in I/O because fewer storage areas need to be accessed. Second, because there is no need to access subsequent partitions, there are now a smaller number of index partitions locked, thus allowing more concurrency. In cases where the next partition is empty, it is possible for more than one partition to be scanned and locked.

Note: Some users may see no change in behavior because the last key value in the index partition may have been beyond the query bounds or, in the case of a unique index definition with an exact match query, a direct key lookup may result as shown below.

```
SQL> SELECT COUNT(*) FROM EMPLOYEES WHERE EMPLOYEE_ID = '00200';
Aggregate      Index only retrieval of relation EMPLOYEES
  Index name  IDX1 [1:1]         Direct lookup
```

The following example shows a partitioned index and three queries. Each query is run in a different process and attaches to the same database.

In previous releases of Oracle Rdb, the first query would lock AREA1 and AREA2 when it only required scanning of AREA1. The second query would then lock AREA2 and AREA_OTHER when it only required scanning of AREA2. Thus, the three queries could not execute concurrently.

The following example demonstrates that a smaller number of index partitions are locked:

```
SQL> CREATE INDEX EMP_INDEX ON EMPLOYEES (EMPLOYEE_ID)
cont> TYPE IS SORTED
cont> STORE USING (EMPLOYEE_ID)
cont>     IN AREA1 WITH LIMIT OF ('00200')
cont>       IN AREA2 WITH LIMIT OF ('00400')
cont>       OTHERWISE IN AREA_OTHER;
SQL>
SQL> -- This query previously locked AREA1 and AREA2.
SQL> -- With the new algorithm, only AREA1 is locked.
SQL>
SQL> DELETE FROM EMPLOYEES WHERE EMPLOYEE_ID < ('00199');
6 rows deleted
SQL>
SQL> -- This query previously locked AREA2 and AREA_OTHER
SQL> -- With the new algorithm, only AREA2 is locked.
SQL>
SQL> DELETE FROM EMPLOYEES WHERE EMPLOYEE_ID > ('00201') AND
cont> EMPLOYEE_ID < ('00399');
5 rows deleted
SQL> -- This query locks AREA_OTHER
SQL> DELETE FROM EMPLOYEES WHERE EMPLOYEE_ID > ('00401');
23 rows deleted
```

The following example demonstrates fewer areas scanned with the new algorithm resulting in less I/O:

```
SQL> CREATE INDEX INDEX_EMP
cont>    ON EMPLOYEES (EMPLOYEE_ID)
cont>    TYPE IS SORTED
cont>    STORE
cont>        USING (EMPLOYEE_ID)
cont>           IN UNIFORM1
cont>                WITH LIMIT OF ('00100')
cont>            IN UNIFORM2
cont>                WITH LIMIT OF ('00200')
cont>            IN UNIFORM3
cont>                WITH LIMIT OF ('00300')
cont>            OTHERWISE IN UNIFORM4;
SQL>
SQL> -- First, delete all employees records in UNIFORM1, UNIFORM2, UNIFORM3
SQL>
SQL> DELETE FROM EMPLOYEES WHERE EMPLOYEE_ID BETWEEN '00001' AND '00300';
12 rows deleted
SQL>
SQL>
SQL> -- Previously, the following  query would result in reading from areas
SQL> -- UNIFORM1, UNIFORM2, UNIFORM3, and UNIFORM4. This occurred because
SQL> -- all partitions were scanned until an index key was found to end the scan.
SQL> -- With the new algorithm, only UNIFORM1 is read, resulting in less I/O.
SQL>
SQL> -- By turning on debug flags (STRATEGY, EXECUTION, INDEX_PARTITION),
SQL> -- the index partitions scanned are displayed.
SQL>
SQL> SET FLAGS 'STRATEGY,EXECUTION,INDEX_PARTITION';
SQL> SELECT * FROM EMPLOYEES WHERE EMPLOYEE_ID = '00020';
~S#0004
Leaf#01 FFirst EMPLOYEES Card=40
  BgrNdx1 INDEX_EMP [1:1] Fan=17
~E#0004.2 Start Area INDEX_EMP.UNIFORM1 (1) <--- ** index partition scanned **
~E#0004.01(1) BgrNdx1 EofData  DBKeys=0  Fetches=0+0  RecsOut=0 #Bufs=0
0 rows selected
```

The same query in previous versions of Rdb7, would result in the empty index partitions being scanned until an index key was found to end the scan.

```
SQL> SET FLAGS 'STRATEGY,EXECUTION,INDEX_PARTITION';
SQL> SELECT * FROM EMPLOYEES WHERE EMPLOYEE_ID = '00020';
~S#0002
Leaf#01 FFirst EMPLOYEES Card=40
  BgrNdx1 INDEX_EMP [1:1] Fan=17
~E#0002.1 Start Area IDX1.UNIFORM1 (1) <--- ** index partitions scanned **
~E#0002.1 Next Area IDX1.UNIFORM2 (2)
~E#0002.1 Next Area IDX1.UNIFORM3 (3)
~E#0002.1 Next Area IDX1.UNIFORM3 (4)
0 rows selected
```

The new algorithm utilizes other data structures to determine that all the data has been returned for the query and eliminates unnecessary index area scans based on the index partition values.

Note

In order to utilize the new index partition scanning algorithm, the logical name RDMS$INDEX_PART_CHECK must be defined to 1. Otherwise, the default is to use the old scanning behavior for partitioned indexes (the same as defining RDMS$INDEX_PART_CHECK = 0 or not defining the logical at all).

This index partition enhancement is not supported for mapped indexes or descending indexes.

9.1.9 UPDATE STATISTICS Clause for ALTER TABLE Statement Implemented for /TYPE=NREL

It is now possible to reacquire table statistics when the ATTACH type is NREL (non-relational DBI gateways). This is done by using the ALTER TABLE UPDATE STATISTICS statement. In prior versions, this was only allowed when the ATTACH type was DBI.

Use of this statement will update the table cardinality and may improve optimization strategies. For example,

```
SQL> ATTACH 'FILE /TYPE=NREL/PATH=PATH-NAME/DICT=DICTIONARY-DRIVER-NAME' ;
SQL> SELECT RDB$CARDINALITY FROM RDB$RELATIONS
cont> WHERE RDB$RELATION_NAME = 'table-name' ;
 RDB$CARDINALITY
               0
1 row selected
SQL> ALTER TABLE table-name UPDATE STATISTICS ;
SQL> SELECT RDB$CARDINALITY FROM RDB$RELATIONS
cont> WHERE RDB$RELATION_NAME = 'table-name' ;
 RDB$CARDINALITY
             322
1 row selected
```

This problem has been corrected in Oracle Rdb7 version 7.0.1.3.

9.1.10 Relaxed Privilege Checking for Temporary Tables

In prior versions of Oracle Rdb7, privileges required for data manipulation operations on global and local temporary tables were the same as those required for base tables. For example, to perform an insert into a global temporary table, a user needed SELECT and INSERT privileges at the database level.

This requirement existed because an insert into a base table implicitly inserts data into the database. The privilege granted at the database level was used to filter the privileges for the table.

However, unlike base tables, the data in temporary tables is not actually stored in the database, thus temporary tables never update the database.

In this release of Oracle Rdb7, only the privileges associated with the temporary table will be considered when performing security validation during data manipulation operations. For example, if the user can attach to the database (requires SELECT privilege only) and is granted INSERT to a global or local temporary table, then the user (or an invokers rights stored routine) will be permitted to update the temporary table. This change will affect the operation of SQL*net for Rdb which no longer requires database manipulation privileges (INSERT, UPDATE, DELETE) for processing temporary tables.

Note

This is a relaxation of the security checking from prior versions of Oracle Rdb7 and only applies to temporary tables.

For previous versions, definers rights stored procedures could be utilized to access the temporary table. The DECLARE LOCAL TEMPORARY TABLE clause generates a "scratch" temporary table which has no associated access control. It is managed by the module which declares it. This type of temporary table is also available through dynamic SQL. This change has been implemented in Oracle Rdb7 Release 7.0.1.3.

9.2 SQL Interface Enhancements

9.2.1 SQL92 Intermediate Level UNIQUE Constraint Available in Rdb7

Oracle Rdb now provides an SQL92 intermediate level compliant UNIQUE constraint. This type of constraint excludes columns which are NULL from the UNIQUE comparison. This effectively allows sets of columns to be UNIQUE or NULL.

This type of constraint will be created by default when the SQL dialect is set to 'SQL89', 'MIA', 'ORACLE LEVEL1' or 'SQL92'. The default dialect is SQLV40. Oracle recommends that you set the dialect to SQL92 (or one of the listed dialects) before using CREATE or ALTER TABLE to add UNIQUE constraints to tables.

Note

The new UNIQUE semantics will be used at run-time under any selected dialect. But the table must be created or altered under the listed dialects to have the new style of unique enabled.

Improved Performance

In addition to conforming to the database language SQL standard (SQL92 Intermediate Level), the new UNIQUE constraint implementation also provides improved performance for single row inserts. This is made possible by eliminating checks for NULL values from the selection expression and thus simplifying the optimization for unique checking.

Here is a comparison of the old and new optimizer strategies. In this example a UNIQUE constraint ("UNIQUE_A") and index on column A is used to check for uniqueness during an INSERT statement. Note that the optimizer chooses a full range search of the index ([0:0]).

```
~S: Constraint "UNIQUE_A" evaluated
Cross block of 2 entries
  Cross block entry 1
    Conjunct        Firstn  Get     Retrieval by DBK of relation T_UNIQUE
  Cross block entry 2
    Conjunct        Aggregate-F2    Conjunct
    Index only retrieval of relation T_UNIQUE
      Index name  T_UNIQUE_INDEX_A [0:0]
```

With the simplified UNIQUE constraint ("UNIQUE_B") the optimizer can use a direct lookup of the index ([1:1]) which reduces the I/O to the index when performing the constraint evaluation.

```
~S: Constraint "UNIQUE_B" evaluated
Cross block of 2 entries
  Cross block entry 1
    Conjunct        Firstn  Get     Retrieval by DBK of relation T_UNIQUE
  Cross block entry 2
    Conjunct        Aggregate-F2    Index only retrieval of relation T_UNIQUE
      Index name  T_UNIQUE_INDEX_B [1:1]
```

Upward Compatibility

In prior versions, the UNIQUE constraint restricted columns to a single NULL value. If you wish to retain this behavior, use the SET DIALECT 'SQLV40' statement before creating new tables or altering existing tables to add UNIQUE constraints.

UNIQUE constraints created in previous versions of Oracle Rdb will still perform as expected. Interfaces such as RDO or the CDD/Repository will continue to define the older style UNIQUE constraint. It is expected that future versions of the Oracle CDD/Repository will implement the new UNIQUE constraint. Database EXPORT and IMPORT will retain the UNIQUE constraint characteristics as defined by the database administrator, regardless of the defined dialect setting.

Note

RMU Extract Item=Table will not distinguish between the old and new UNIQUE constraints in this release of Rdb. The generated SQL script must be modified to establish the appropriate dialect before using it to create a database.

Because this new style of UNIQUE constraint is a relaxation of the UNIQUE rules, it is possible to drop the old style UNIQUE constraint and redefine the constraint under the SQL92 dialect.

Note that this meaning of UNIQUE (excluding NULL from the uniqueness test) does not apply to the UNIQUE index which still does not allow duplicate entries for NULL. If a UNIQUE index is currently defined which assists the UNIQUE constraint optimization, the database administrator may wish to drop the index and make it a non-UNIQUE index so that multiple NULLs can be stored. The UNIQUE constraint will still enforce the uniqueness of the data.

You can use the SQL SHOW TABLE command to determine which type of UNIQUE constraint is in use. The following example shows a UNIQUE constraint created when the default dialect was used (SQLV40). A new description follows the "Unique constraint" text, explaining the interpretation of null values.

```
SQL> SHOW TABLE (CONSTRAINT) T_UNIQUE
Information for table T_UNIQUE

Table constraints for T_UNIQUE:
T_UNIQUE_UNIQUE_B_A
 Unique constraint
     Null values are considered the same
 Table constraint for T_UNIQUE
 Evaluated on UPDATE, NOT DEFERRABLE
 Source:
        UNIQUE (b,a)
   .
   .
   .
```

The next example shows a UNIQUE constraint created when the dialect was set to 'SQL92', and the description here indicates that all null values are considered distinct.

```
SQL> SHOW TABLE (CONSTRAINT) T_UNIQUE2
Information for table T_UNIQUE2

Table constraints for T_UNIQUE2:
T_UNIQUE2_UNIQUE_B_A
 Unique constraint
     Null values are considered distinct
 Table constraint for T_UNIQUE2
 Evaluated on UPDATE, NOT DEFERRABLE
 Source:
        UNIQUE (b,a)
   .
   .
   .
```

Additional Constraint Improvements

As a side effect of this change, Oracle Rdb also recognizes a larger class of CHECK constraints as being uniqueness checks. The main benefit is that these constraints are no longer processed when a DELETE is executed for the table, because DELETE does not affect the uniqueness of the remaining rows.

The following is an example of this CHECK constraint:

```
SQL> CREATE TABLE T_USER_UNIQUE_NEW (
cont>     A INTEGER,
cont>     B INTEGER,
cont>     CONSTRAINT UNIQUE_AB_NEW
cont>         CHECK ((SELECT COUNT(*)
cont>                 FROM T_USER_UNIQUE_NEW t2
cont>                 WHERE T2.A = T_USER_UNIQUE_NEW.A AND
cont>                       T2.B = T_USER_UNIQUE_NEW.B) <= 1)
cont>             NOT DEFERRABLE
cont>     );
```

In previous versions of Rdb only equality with 1 was recognized as a uniqueness constraint. In this example, a comparison of LESS THAN or EQUAL to one also qualifies as a uniqueness constraint.

9.2.2 Enhancements to DROP STORAGE AREA ... CASCADE

Oracle Rdb7 Release 7.0.1.3 contains several corrections and enhancements to the DROP STORAGE AREA ... CASCADE feature.

DROP INDEX ... CASCADE is Performed if Whole Index is in a Single Area

In previous releases, the DROP STORAGE AREA ... CASCADE command would fail if a partitioned table had an index which was not partitioned and it resided entirely in the storage area being dropped.

This restriction has been removed. Now the index itself will be dropped using CASCADE semantics and this will invalidate any query outlines that reference the index.

Not all Constraints are Evaluated by DROP STORAGE AREA ... CASCADE

The NOT NULL, PRIMARY KEY, and UNIQUE constraints for affected tables are ignored by DROP STORAGE AREA ... CASCADE in this release, because validation of these constraints is not warranted.

These types of constraints are not affected by the removal of rows from a table. This can save considerable I/O and elapsed time when performing the DROP STORAGE AREA ... CASCADE command. However, CHECK and FOREIGN constraints on the affected table, and referencing tables, will still be evaluated.

Debugging Output now Available for DROP STORAGE AREA ... CASCADE

When the DROP STORAGE AREA ... CASCADE command is executing, it will log debugging messages to the standard output device (SYS$OUTPUT) or the RDMS$DEBUG_FLAGS_OUTPUT log file, if defined.

This logging can be enabled using the new logical name RDMS$SET_FLAGS which accepts the same input as the SQL SET FLAGS statement.

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

These SET FLAGS options enable the following debug output:

- STOMAP_STATS will display the processing of storage maps for any tables which reference the dropped storage area. The output will be prefixed with "~As". This is identical to using RDMS$DEBUG_FLAGS defined as "As".
- INDEX_STATS will display the processing of any indexes which reference the dropped storage area. The output will be prefixed with "~Ai". This is identical to using RDMS$DEBUG_FLAGS defined as "Ai".
- ITEM_LIST will display the names of any constraints that require processing. This is identical to using RDMS$DEBUG_FLAGS defined as "H".

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

As part of the DROP STORAGE AREA ... CASCADE operation, tables and indexes may be deleted. These are processed internally as DROP TABLE ... CASCADE and DROP INDEX ... CASCADE operations. However, by the time these commands execute, all references to the dropped storage area will have been removed so, in many cases, the DROP simply cleans up the metadata definition and need not scan the storage area.

In the following example it can be seen that a single DROP STORAGE AREA ... CASCADE operation needs to scan four logical areas to destroy the hash indexes (see "destroy hash" in the example). The scanning of an area takes I/O and time and should be avoided if possible.

```
SQL> ALTER DATABASE
cont>     FILENAME 'TEST_MFDB'
cont>     DROP STORAGE AREA S_AREA_1A CASCADE;
~As: Drop Storage Area "S_AREA_1A" Cascade
~As: ...area referenced by map: "SR_MAP"
~As: ...area referenced by map: "PV_MAP"
~As: ...area referenced by map: "S_MAP"
~As: ...area referenced by map: "SF_MAP"
~As: ...area referenced by index: "SR_1H"
~As: ...area referenced by index: "PV_2H"
~As: ...area referenced by index: "S_1H"
~As: ...area referenced by index: "SF_1H"
~As: ...update the AIP for larea=64 (table)
~As: ...update the AIP for larea=65 (table)
~As: ...update the AIP for larea=66 (table)
~As: ...update the AIP for larea=67 (table)
~As: ...update the AIP for larea=56 (index)
~As: ...update the AIP for larea=58 (index)
~As: ...update the AIP for larea=60 (index)
~As: ...update the AIP for larea=62 (index)
~As: ...update the AIP for larea=47 (sysrec)
~As: ...drop table "SF" cascade
~Ai delete index (cascade) SF_2H
        destroy Hash index, Idx=57, Sys=48
~Ai delete index (cascade) SF_1H
~As: ...drop table "S" cascade
~Ai delete index (cascade) S_4H
        destroy Hash index, Idx=59, Sys=50
~Ai delete index (cascade) S_1H
~As: ...drop table "PV" cascade
~Ai delete index (cascade) PV_4H
        destroy Hash index, Idx=61, Sys=51
~Ai delete index (cascade) PV_2H
~As: ...drop table "SR" cascade
~Ai delete index (cascade) SR_2H
        destroy Hash index, Idx=63, Sys=49
~Ai delete index (cascade) SR_1H
~As: ...4 logical areas were scanned in other areas
~As: ...Reads: async 477 synch 103, Writes: async 144 synch 22
```

This revised script drops several areas in a specific order so that no logical area scans are performed. Even for this simple example database, the read/write I/O statistics (on the last line of each log) can be compared to see the improvement.

```
SQL> ALTER DATABASE
cont>     FILENAME 'TEST_MFDB'
cont>     DROP STORAGE AREA SF_AREA_1A CASCADE
cont>     DROP STORAGE AREA S_AREA_4A CASCADE
cont>     DROP STORAGE AREA PV_AREA_4A CASCADE
cont>     DROP STORAGE AREA SR_AREA_1A CASCADE
cont>     DROP STORAGE AREA S_AREA_1A CASCADE;
~As: Drop Storage Area "SF_AREA_1A" Cascade
~As: ...area referenced by index: "SF_2H"
~As: ...dropping index "SF_2H" (not partitioned)
~As: ...update the AIP for larea=57 (index)
~As: ...update the AIP for larea=48 (sysrec)
~As: ...drop index "SF_2H" cascade
~Ai delete index SF_2H               (1)
~As: ...Reads: async 0 synch 15, Writes: async 11 synch 4
~As: Drop Storage Area "S_AREA_4A" Cascade
~As: ...area referenced by index: "S_4H"
~As: ...dropping index "S_4H" (not partitioned)
~As: ...update the AIP for larea=59 (index)
~As: ...update the AIP for larea=50 (sysrec)
~As: ...drop index "S_4H" cascade
~Ai delete index S_4H                    (1)
~As: ...Reads: async 0 synch 1, Writes: async 0 synch 7
~As: Drop Storage Area "PV_AREA_4A" Cascade
~As: ...area referenced by index: "PV_4H"
~As: ...dropping index "PV_4H" (not partitioned)
~As: ...update the AIP for larea=61 (index)
~As: ...update the AIP for larea=51 (sysrec)
~As: ...drop index "PV_4H" cascade
~Ai delete index PV_4H           (1)
~As: ...Reads: async 0 synch 2, Writes: async 0 synch 17
~As: Drop Storage Area "SR_AREA_1A" Cascade
~As: ...area referenced by index: "SR_2H"
~As: ...dropping index "SR_2H" (not partitioned)
~As: ...update the AIP for larea=63 (index)
~As: ...update the AIP for larea=49 (sysrec)
~As: ...drop index "SR_2H" cascade
~Ai delete index SR_2H                   (1)
~As: ...Reads: async 0 synch 0, Writes: async 0 synch 18
~As: Drop Storage Area "S_AREA_1A" Cascade
~As: ...area referenced by map: "SR_MAP"
~As: ...area referenced by map: "PV_MAP"
~As: ...area referenced by map: "S_MAP"
~As: ...area referenced by map: "SF_MAP"
~As: ...area referenced by index: "SR_1H"
~As: ...area referenced by index: "PV_2H"
~As: ...area referenced by index: "S_1H"
~As: ...area referenced by index: "SF_1H"
~As: ...update the AIP for larea=64 (table)
~As: ...update the AIP for larea=65 (table)
~As: ...update the AIP for larea=66 (table)
~As: ...update the AIP for larea=67 (table)
~As: ...update the AIP for larea=56 (index)
~As: ...update the AIP for larea=58 (index)
~As: ...update the AIP for larea=60 (index)
~As: ...update the AIP for larea=62 (index)
~As: ...update the AIP for larea=47 (sysrec)
~As: ...drop table "SF" cascade
~Ai delete index (cascade) SF_1H
~As: ...drop table "S" cascade
~Ai delete index (cascade) S_1H
~As: ...drop table "PV" cascade
~Ai delete index (cascade) PV_2H
~As: ...drop table "SR" cascade
~Ai delete index (cascade) SR_1H
```

```
~As: ...Reads: async 0 synch 55, Writes: async 96 synch 32
```

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

Note that the read/write I/O statistics are only output if the database has statistics collection enabled. Statistics collection may be disabled when the logical name RDM$BIND_STATS_ENABLED is assigned the value 0, or in the database using the ALTER DATABASE ... STATISTICS COLLECTION IS DISABLED command.

9.2.3 New SQL SET FLAGS Options

New keywords for the SET FLAGS statement

This release of Oracle Rdb7 adds new keywords for use by the SET FLAGS statement and the RDMS$SET_FLAGS logical name. The keywords are not case sensitive and can be abbreviated to any unambiguous prefix.

Table 9–1 Rdb Flag Keywords

Keyword	Negated Keyword	Debug Flags Equivalent	Comment
COSTING	NOCOSTING[1]	Oc	Displays traces on optimizer costing
CURSOR_STATS	NOCURSOR_STATUS[1]	Og	Displays general cursor statistics for optimizer
INDEX_COLUMN_ GROUP	NOINDEX_COLUMN_ GROUP[1]	n/a	Enables leading index columns as workload column group. This may increase solution cardinality accuracy
SOLUTIONS	NOSOLUTIONS[1]	Os	Displays traces on optimizer solutions
TRANSITIVITY[1]	NOTRANSITIVITY	RDMS$DISABLE_ TRANSITIVITY	Enables transitivity between selections and join predicates
MAX_STABILITY	NOMAX_STABILITY[1]	RDMS$MAX_ STABILITY	Enables maximum stability (dynamic optimizer not allowed)
OLD_COST_MODEL	NOOLD_COST_ MODEL[1]	RDMS$USE_ OLD_COST_ MODEL	Enables old cost model
REVERSE_SCAN[1]	NOREVERSE_SCAN	RDMS$DISABLE_ REVERSE_ SCAN	Enables reverse index scan strategy.
ZIGZAG_MATCH[1]	NOZIGZAG_MATCH	RDMS$DISABLE_ ZIGZAG_ MATCH	Enables zigzag key skip on both outer and inner match loops.[2]

[1]Default value

[2]ZIGZAG_MATCH, NOZIGZAG_OUTER disables zigzag key skip on outer loop (equivalent to defining the logical name RDMS$DISABLE_ZIGZAG_MATCH to a value of 1). NOZIGZAG_MATCH disables zigzag key skip on both outer and inner match loops (equivalent to defining the logical name RDMS$DISABLE_ZIGZAG_MATCH to a value of 2)

(continued on next page)

Table 9–1 (Cont.) Rdb Flag Keywords

Keyword	Negated Keyword	Debug Flags Equivalent	Comment
ZIGZAG_OUTER[1]	NOZIGZAG_OUTER	RDMS$DISABLE_ ZIGZAG_ MATCH	Enables zigzag key skip on outer loop.[2]

[1]Default value

[2]ZIGZAG_MATCH, NOZIGZAG_OUTER disables zigzag key skip on outer loop (equivalent to defining the logical name RDMS$DISABLE_ZIGZAG_MATCH to a value of 1). NOZIGZAG_MATCH disables zigzag key skip on both outer and inner match loops (equivalent to defining the logical name RDMS$DISABLE_ZIGZAG_MATCH to a value of 2)

New logical name RDMS$SET_FLAGS

The new logical name RDMS$SET_FLAGS accepts a string in the same format as provided to the SQL SET FLAGS statement. Abbreviations, values and negation (NO) of keywords are also supported. The equivalence string is processed after the logical name RDMS$DEBUG_FLAGS during attach to the database. Therefore, settings in RDMS$DEBUG_FLAGS will be superseded by keywords defined by this logical name. Unlike other Oracle Rdb logical names, an exception is raised if an error is found in the RDMS$SET_FLAGS string and the attach to the database will fail.

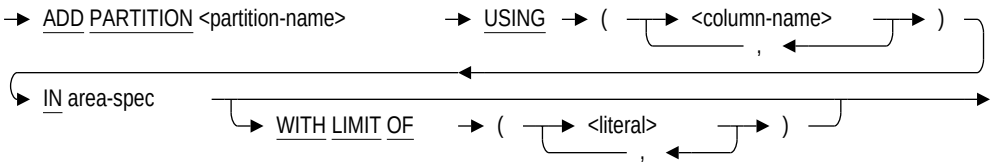
The SQL SHOW FLAGS command can be used to see which flags are set during an interactive SQL session.

9.2.4 New ADD PARTITION Clause for ALTER INDEX

The ALTER INDEX command has been enhanced with this release of Oracle Rdb7.

A new ADD PARTITION clause is now available to add a single partition within an existing HASHED index.

add-partition-clause =



Usage Notes

- The partition name is currently ignored by Oracle Rdb7. In a future release Rdb will store the name in the system table RDB$STORAGE_MAP_AREAS so that it can be used with other partition related statements. The name will then be validated and must be unique per index.
- ADD PARTITION is currently only supported for hashed indexes. Support for sorted indexes will be provided in a future release.
- The index must have been created with a STORE clause, so that additional partitions can be added.

- There must be no active queries compiled against this table. This includes declared cursors in the current session, or other applications which have referenced the table. As with other ALTER INDEX statements exclusive access to the table is required during the current transaction.
- The USING clause must list the same column names in the same order as in the original index definition.
- If no WITH LIMIT OF clause is specified then the partition will be added at the end of the index as an OTHERWISE partition. If there is an existing OTHERWISE partition for this index then an error will be reported.
- When a new final partition or an OTHERWISE partition is successfully added, no I/O to the index is required. That is, no data in the index needs to be relocated.
- The WITH LIMIT OF clause must specify a new unique set of values for the partition. There must exist a literal value for each column listed in the USING clause.

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

  Note

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

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

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

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

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

If the logical name RDMS$CREATE_LAREA_NOLOGGING is defined then the hash buckets and duplicate nodes written to the new partition will not be journaled. However, the updates to the existing RDB$SYSTEM_RECORD in that partition, and the deletes performed on the following partition will be journaled.

- If the INDEX_STATS flag is enabled then the ALTER INDEX command will then log messages to the RDMS$DEBUG_FLAGS_OUTPUT file (or SYS$OUTPUT if not defined) reporting the progress of the ADD PARTITION clause. INDEX_STATS can be enabled using the SET FLAGS 'INDEX_STATS' command or by defining the RDMS$DEBUG_FLAGS logical to "Ai" (with a lower case i). See the example below in Example 9–2.

Note

The read/write I/O statistics shown in the example are not displayed if STATISTICS COLLECTION IS DISABLED on the database, or if the logical name RDM$BIND_STATS_ENABLED is defined to 0.

- The SHOW INDEX, or SHOW TABLE (INDEX) command will display the original source of the index definition, with the ADD PARTITION source appended. See the example below in Example 9–4. Use RMU/EXTRACT /ITEM=INDEX to see the current index definition with the additional partitions merged into the SQL CREATE INDEX syntax.

Examples

The example below use an index definition as shown in Example 9–1. Example 9–2 shows the syntax for adding a partition before the final partition of an index.

Example 9–1 Original Index Definition

```
SQL> CREATE UNIQUE INDEX EMPLOYEES_INDEX
cont>     ON EMPLOYEES (EMPLOYEE_ID)
cont>     TYPE IS HASHED
cont>     STORE USING (EMPLOYEE_ID)
cont>         IN JOBS WITH LIMIT OF ('00999');
```

This requires that the final partition (which now follows the new partition) be scanned and matching keys moved to the new partition.

Example 9–3 shows the syntax for adding a partition after the final partition of an index. This required no I/O to the partition because there is no following partition and therefore no keys to be moved.

Example 9–4 shows the output from SHOW INDEX with the ADD PARTITION syntax appended to the original source of the index.

Example 9–2 Adding a Partition Before the Final Partition

```
SQL> SET TRANSACTION READ WRITE
cont>   RESERVING EMPLOYEES for EXCLUSIVE WRITE;
SQL> ALTER INDEX EMPLOYEES_INDEX
cont>     ADD PARTITION NEW_EMPS_200
cont>       USING (EMPLOYEE_ID)
cont>       IN EMP_INFO WITH LIMIT OF ('00200');
~Ai alter index "EMPLOYEES_INDEX" (hashed=1, ordered=0)
~Ai add partition "NEW_EMPS_200" : area "EMP_INFO"
~Ai storage area "EMP_INFO" larea=121
~Ai splitting partition #1
~Ai split complete: total 100 keys, moved 37 (dups 0)
~Ai reads: async 8 synch 16, writes: async 22 synch 0
SQL> COMMIT;
```

Example 9–3 Adding a New Final Partition

```
SQL> SET TRANSACTION READ WRITE
cont>   RESERVING EMPLOYEES FOR EXCLUSIVE WRITE;
SQL> ALTER INDEX EMPLOYEES_INDEX
cont>     ADD PARTITION NEW_EMPS_1400
cont>       USING (EMPLOYEE_ID)
cont>       IN EMPIDS_OVER WITH LIMIT OF ('01400');
~Ai alter index "EMPLOYEES_INDEX" (hashed=1, ordered=0)
~Ai add partition "NEW_EMPS_1400" : area "EMPIDS_OVER"
~Ai storage area "EMPIDS_OVER" larea=122
~Ai adding new final partition 3
SQL> COMMIT;
```

Example 9–4 Adding a Partition Before the Final Partition

```
SQL> SHOW INDEX EMPLOYEES_INDEX
Indexes on table EMPLOYEES:
EMPLOYEES_INDEX                 with column EMPLOYEE_ID
  No Duplicates allowed
  Type is Hashed Scattered
  Compression is DISABLED
Store clause:           STORE using (EMPLOYEE_ID)
                             in JOBS with limit of ('00999')
                        Add Partition partition NEW_EMPS_200
                        using (EMPLOYEE_ID)
                        in EMP_INFO with limit of ('00200')
                        Add Partition partition NEW_EMPS_1400
                        using (EMPLOYEE_ID)
                        in EMPIDS_OVER with limit of ('01400')
```

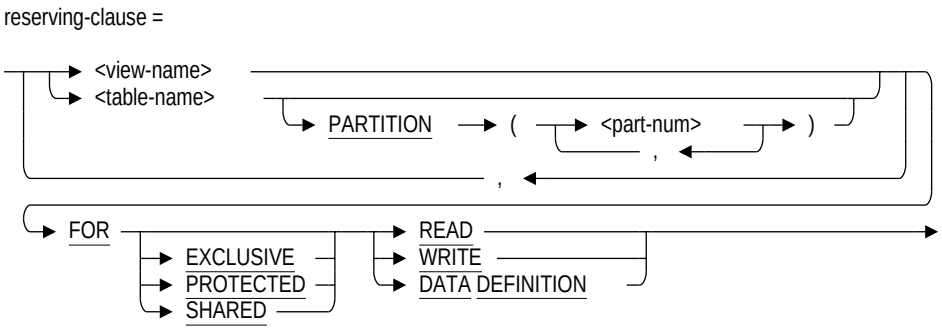
9.2.5 Enhancement to the SET TRANSACTION Statement

Bug 548039.

The SET TRANSACTION and DECLARE TRANSACTION statements have been enhanced for Oracle Rdb7 Release 7.0.1.3 so that selected partitions of a horizontally partitioned table can be independently reserved.

The objective is to allow concurrent partitioned operations on a single table with the highest locking modes available.

Figure 9–1 RESERVING Clause



Syntax

The changed syntax for the RESERVING clause show in Figure 9–1.

The **part-num** is the number for the partition to be reserved, or locked. Only the values in the RDB$STORAGE_MAP_AREAS table in the RDB$ORDINAL_POSITION column may be specified. Duplicate **part-num** values in the RESERVING clause will be ignored by SQL. Access to partitions not listed in the reserving clause will default to SHARED access.

The PARTITION clause is not permitted if a table is not mapped (has no storage map), or has a map which is vertically partitioned (uses the STORE COLUMNS clause). If an index has an identical STORE clause as the storage map then it will also be locked using the same list of partition numbers.

```
SQL> SET TRANSACTION READ WRITE
cont> RESERVING EMPLOYEES PARTITION (2) FOR EXCLUSIVE WRITE;
```

In this example just the second partition will be locked in EXCLUSIVE WRITE mode. The advantage is that this process can now insert, update or delete from this partition without writing to the snapshot file (.snp), and in general uses less resources for operations on the partition. Several processes can now concurrently update the EMPLOYEES table (providing each uses a distinct set of partitions) and use EXCLUSIVE access.

Customers should be advised that using the PARTITION clause needs careful database and application design. For instance, if the indices are partitioned using different partitioning keys, or different value ranges then cross partition updates could lead to deadlocks and other lock conflicts between the concurrent update processes.

Note

The PARTITION clause is not compatible with the DATA DEFINITION clause.

9.2.6 Computed Column Restriction Lifted for CREATE TRANSFER

Bug 572514.

Until now, SQL has imposed a restriction on the definitions of computed columns used in CREATE TRANSFER statements. The computed column definitions were not permitted to refer to domain names. If such column definitions were encountered, SQL issued the following warning message.

```
"SQL$_CMPBYWNRL, Invalid computed field <column-name> will not be
transferred from relation <table-name>"
```

That column would then be removed from the list of those to be transferred.

This restriction has been removed from SQL. Removal of this restriction in SQL, however, does not completely solve the problem. If you attempt to create and execute a transfer without taking preparatory steps (see workaround farther on), execution of the transfer will fail if you are using the Replication Option for Rdb release 7.0.1 or earlier. Those versions of the Replication Option are not able to transfer the definitions of domains referenced only within computed columns.

The following example shows domain and table definitions and a CREATE TRANSFER statement which would have resulted in the SQL$_CMPBYWNRL warning message from SQL.

```
SQL> ATTACH 'FILE DISK:[DIR]SOURCE.RDB';
SQL>
SQL> -- Create a table with two columns, one of which is computed and whose
SQL> -- definition references the name of a domain.
SQL>
SQL> CREATE DOMAIN DOM1 SMALLINT;
SQL>
SQL> CREATE TABLE TAB1 (
cont> COL1 INTEGER,
cont> COL2 COMPUTED BY
cont>   CAST(SUBSTRING(CAST(COL1 AS CHAR(4)) FROM 1 FOR 2) AS DOM1));
SQL>  COMMIT;
SQL>
SQL> -- Prior to lifting the restriction in SQL, the following transfer definition
SQL> -- would have resulted in a SQL warning message:  %SQL-W-CMPBYWNRL, Invalid
SQL> -- computed field COL2 will not be transferred from relation TAB1.
SQL>
SQL>  CREATE TRANSFER COMPUTED_DOMAIN_REF TYPE IS EXTRACTION
cont>   MOVE TABLES TAB1
cont>   TO EXISTING FILENAME DISK:[DIR]TARGET.RDB
cont>   LOGFILE IS DISK:[DIR]COMPUTED_DOMAIN_REF.LOG;
```

To successfully perform this transfer using a version of the Replication Option for Rdb which does not transfer domains referenced within computed columns, use the following workaround. In the preceding example, using the new version of SQL, the transfer definition resulting from the CREATE TRANSFER statement would include the COL2 column to be transferred. Since the DOM1 domain is only referenced within the definition of COL2, a computed column, the Replication Option does not recreate that DOM1 definition in the target database. Therefore, prior to the first execution of the transfer, you must add the DOM1 definition to the target database yourself, using a CREATE DOMAIN statement as shown in the preceding example.

The restriction on the use of domain references within computed columns used in a CREATE TRANSFER statement has been removed from SQL in Oracle Rdb7 Release 7.0.1.3.

9.2.7 Change In Functionality for RESTRICTED ACCESS Clause

A transaction which reserves a table for EXCLUSIVE access does not also reserve the LIST area for EXCLUSIVE access. The LIST (segmented string) area is usually shared by many tables and therefore SHARED access is assumed, by default, to permit updates to the other tables.

This means that during an RMU/LOAD operation or an application update of a table reserved for EXCLUSIVE access, it may be observed that the snapshot storage area (.snp) grows. This is due to the I/O to the LIST area which is performed by default using SHARED WRITE mode.

In the original release of Oracle Rdb7, the RESTRICTED ACCESS clause on the ATTACH statement was changed so that all storage areas were accessed in EXCLUSIVE mode. This clause should be used to eliminate the snapshot I/O and related overhead when performing a lot of I/O to the LIST storage areas, such as when restructuring the database or dropping a large table containing LIST OF BYTE VARYING columns and data.

Note

RESTRICTED ACCESS is the default for SQL IMPORT, therefore, there is reduced overhead during the IMPORT of LIST data.

9.2.8 SQL Expression Support for ORDER BY and GROUP BY Clauses

Until now SQL syntax prohibited the use of expressions in either the ORDER BY or GROUP BY clauses. Now expressions are supported in both places. Note the following restrictions when using GROUP BY expressions.

- You must have a syntactically similar expression in the select list.
- The star (*) is not supported when using expressions with GROUP BY.
- GROUP BY expressions are not supported in subqueries.

The following platforms are affected by this feature:

- Interactive SQL
- SQL module language
- Precompiled SQL

The following examples show both proper and improper uses of expressions with ORDER BY and GROUP BY.

```
SQL> SELECT * FROM X ORDER BY ABS(XCOL1 - 3);
       XCOL1         XCOL2
           2            10
           1             1
           6           100
3 rows selected
SQL> SELECT (XCOL1 + 2) COL FROM X GROUP BY (XCOL1 + 2);
                   COL
                     3
                     4
                     8
3 rows selected
SQL> SELECT (2 + XCOL1) COL FROM X GROUP BY (XCOL1 + 2);
%SQL-F-NOTGROFLD, Column XCOL1 cannot be referred to in the select
list, ORDER BY, or HAVING clause because it is not in the GROUP BY clause
SQL> SELECT * FROM X GROUP BY (XCOL1 + 2);
%SQL-F-INVSELSTAR, * is not allowed in this context
```

9.3 Oracle RMU Enhancements

9.3.1 [No]Commit Qualifier Added to RMU/RESTORE Command

A new qualifier, [No]Commit, has been added to the RMU/RESTORE command. This qualifier is only used when the backup file being restored is from a previous version of Oracle Rdb. Explicitly specifying the COMMIT qualifier instructs RMU to commit the converted database to the current version of Oracle Rdb before completing the restoration. In this case, the conversion is permanent and the database cannot be returned to the previous version. This is also the default behavior if the COMMIT qualifier is not used. Specifying NOCOMMIT instructs RMU not to commit the converted database. In this case, the database may later be rolled back to its original version using the RMU/CONVERT ROLLBACK command or it may be permanently committed to the current version using the RMU/CONVERT COMMIT command.

9.3.2 /WAIT Qualifier Added to RMU/OPEN Command

Previously, the RMU/OPEN command could return before a database was completely open and available. This was generally most obvious when a database was re-opened after a node failure and the database recovery processes ran for a long time.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.3. A /WAIT qualifier has been added to the RMU/OPEN command. When /WAIT is specified, the RMU/OPEN command will not return until the database is open and completely recovered. At this point, the database is available for normal access. The default behavior is /NOWAIT which is the same behavior that database open has always had (the RMU/OPEN command returns before recovery completes).

9.3.3 Limit the Number and Size of AIJ Initialization I/O Buffers

All OpenVMS platforms.

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

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

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

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

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

9.3.4 RMU/SHOW SYSTEM and RMU/SHOW USERS Now Include Elapsed Times

The Oracle Rdb RMU/SHOW SYSTEM and RMU/SHOW USERS commands now display elapsed as well as absolute times for the time that the monitor was started and the time that databases were opened.

The following example demonstrates this output:

```
$ RMU/SHOW USERS
Oracle Rdb V7.0-13 on node HOTRDB  2-APR-1998 16:56:05.43
    - monitor started  1-APR-1998 16:51:09.37 (uptime 1 00:04:56.06)
    - monitor log filename is "DISK$:[RDM$MONITOR]RDMMON70.LOG;1"

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

9.3.5 New Restricted_Access Qualifier for RMU/LOAD

The RMU/LOAD command now supports the RESTRICTED_ACCESS option when attaching to an Oracle Rdb database. This option allows a single process to load data and enables some optimizations available only when RESTRICTED ACCESS is in use.

If you are loading a table from an RMU Unload file which contains LIST OF BYTE VARYING data, the /RESTRICTED_ACCESS option will reserve the LIST areas for EXCLUSIVE access. This reduces the virtual memory used by long transactions in RMU Load, and also eliminates I/O to the snapshot files for the LIST storage areas.

The RESTRICTED_ACCESS and PARALLEL options are mutually exclusive and may not both be specified on the RMU Load command line, or within a plan file. While RMU Load is running with this option enabled, no other user may attach to the database. The default is NORESTRICTIED_ACCESS.

9.3.6 New Qualifier for RMU/SHOW STATISTICS Command

The RMU/SHOW STATISTICS utility consumes approximately 13 thousand bytes of virtual memory per logical area. Also, the number of logical areas is determined by the largest logical area identifier, not the actual number of areas.

This can result in the RMU/SHOW STATISTICS utility consuming large amounts of virtual memory, even if you do not wish to review logical area statistic information.

There currently is no method available to disable the display of logical area statistic information.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.3. A new qualifier for the RMU/SHOW STATISTICS command, **/[NO]LOGICAL_AREA**, can be used to indicate that you do not wish to display logical area statistics information. By specifying the /NOLOGICAL_AREA qualifier, the virtual memory for logical area statistics information presentation will not be acquired.

Be careful when specifying the /NOLOGICAL_AREA qualifier that you do not specify **/NOLOG**, which will cause logical area statistic information to still be collected.

The command default is **/LOGICAL_AREA**.

There is no corresponding configuration variable. This qualifier cannot be modified at run-time.

9.3.7 RMU/SHOW STATISTICS "Automatic Screen Capture" Facility

The **RMU/SHOW STATISTICS** utility has been enhanced to provide an "Automatic Screen Capture" facility. This facility allows you to automatically capture images of all screens, at a specified interval. The facility is similar to using the "Options" onscreen-menu option every so often.

The "Automatic Screen Capture" facility is invoked using the "Start automatic screen capture" option of the "Tools" menu (obtained using the "!" keystroke). You will be requested to enter the interval *between* screen capture operations, expressed in seconds. The minimum interval is 30 seconds.

It takes approximately 5 to 10 seconds to capture all available screens. You will be notified when the screens are being captured by the message "*** Writing Report ***" being displayed on the status region of the current screen.

In order to guarantee consistent statistic information, statistic information updates are temporarily "paused" while the screen capture operation is occurring. Note that this "pause" also affects writing to the binary output file, as well as any log files being recorded.

The "Automatic Screen Capture" facility can be disabled using the "Stop automatic screen capture" option of the "Tools" menu.

The "Automatic Screen Capture" facility can also be invoked using the configuration variable **REPORT_INTERVAL** which specifies the number of seconds.

There is no command qualifier for this facility. Also, you cannot use the facility if the **/NOINTERACTIVE** qualifier is specified.

The "Automatic Screen Capture" facility works with binary files.

The "Automatic Screen Capture" facility is integrated with the cluster statistic collection facility. If cluster statistic collection is enabled, all supported screens will provide cluster information.

9.3.8 RMU/SHOW STATISTIC "Logical Area Overview" Screen

The RMU/SHOW STATISTIC utility has been enhanced to provide a "Logical Area Overview" screen. Located in the "Logical Area Information" sub-menu, the logical area overview screen provides a comparison of all logical areas of a particular type.

The following is an example of the "Logical Area Overview" screen:

```
Node: ALPHA3 (1/1/16)   Oracle Rdb X7.0-00 Perf. Monitor 18-MAR-1998 14:20:54.98
Rate: 1.00 Second        Logical Area Overview (Tables)     Elapsed: 03:28:56.70
Page: 1 of 1    KODH$:[R_ANDERSON.WORK.STATS]MF_PERSONNEL.RDB;1     Mode: Online
------------------------------------------------------------------------------
Logical.Area.Name... record fetch record store record erase    discarded CurTot
RDB$RELATIONS.RDB$SY           29            0            0            0
RDB$FIELD_VERSIONS.R          217            0            0            0
RDB$INDICES.RDB$SYST           35            0            0            0
RDB$INDEX_SEGMENTS.R           35            0            0            0
RDB$FIELDS.RDB$SYSTE           12            0            0            0
RDB$RELATION_FIELDS.           12            0            0            0
RDB$DATABASE.RDB$SYS            1            0            0            0
RDB$VIEW_RELATIONS.R            0            0            0            0
RDB$CONSTRAINT_RELAT            6            0            0            0
RDB$CONSTRAINTS.RDB$            6            0            0            0
RDB$STORAGE_MAPS.RDB            9            0            0            0
RDB$STORAGE_MAP_AREA           17            0            0            0
RDB$INTERRELATIONS.R            0            0            0            0
RDB$COLLATIONS.RDB$S            0            0            0            0
RDB$TRIGGERS.RDB$SYS            1            0            0            0
RDB$RELATION_CONSTRA            0            0            0            0
RDB$RELATION_CONSTRA            0            0            0            0
RDB$PRIVILEGES.RDB$S            0            0            0            0
RDB$MODULES.RDB$SYST            0            0            0            0
RDB$ROUTINES.RDB$SYS            0            0            0            0
RDB$PARAMETERS.RDB$S            0            0            0            0
RDB$QUERY_OUTLINES.R            0            0            0            0
RDB$WORKLOAD.RDB$SYS           37            0            0            0
CANDIDATES.RDB$SYSTE            0            0            0            0
COLLEGES.EMP_INFO               0            0            0            0
DEGREES.EMP_INFO              495            0          165            0
DEPARTMENTS.DEPARTME         2626            0            0            0
EMPLOYEES.EMPIDS_LOW          148            0           37            0
EMPLOYEES.EMPIDS_MID          228            0           57            0
EMPLOYEES.EMPIDS_OVE           24            0            6            0
JOBS.JOBS                       0            0            0            0
JOB_HISTORY.EMPIDS_L          306            0          102            0
JOB_HISTORY.EMPIDS_M          450            0          150            0
JOB_HISTORY.EMPIDS_O           66            0           22            0
RESUMES.EMP_INFO            58600            0            0            0
SALARY_HISTORY.SALAR         2187            0          729            0
WORK_STATUS.EMP_INFO            0            0            0            0

------------------------------------------------------------------------------
Config Exit Help Menu >next_page <prev_page Options Pause Reset Set_rate Write
```

The "Logical Area Overview" screen displays the following information:

- **Logical Area Name**. This column displays the name of the logical area, followed by a period ("."), followed by the name of the physical area (storage area) in which the logical area partition resides.

A maximum of 20 characters is displayed, which typically results in the storage area name being truncated.

For performance reasons, the logical area names are *not* sorted in any particular order.

- **Statistic Field #1**. This column displays a user-selectable statistic field appropriate for the logical area type.

  The default statistic field is the following:

  - Table - record fetch.
  - B-tree Index - leaf fetches.
  - Hash Index - hash index fetched.
  - Blob - blob fetched.

- **Statistic Field #2**. This column displays a user-selectable statistic field appropriate for the logical area type.

  The default statistic field is the following:

  - Table - record stored.
  - B-tree Index - leaf insertion.
  - Hash Index - hash insertion.
  - Blob - blob stored.

- **Statistic Field #3**. This column displays a user-selectable statistic field appropriate for the logical area type.

  The default statistic field is the following:

  - Table - record erased.
  - B-tree Index - leaf removal.
  - Hash Index - hash deletion.
  - Blob - blob erased.

- **Statistic Field #4**. This column displays a user-selectable statistic field appropriate for the logical area type.

  The default statistic field is the following:

  - Table - pages discarded.
  - B-tree Index - pages discarded.
  - Hash Index - pages discarded.
  - Blob - pages discarded.

- **Statistic Type**. This column identifies the "type" of statistic information being displayed. The following types are available:

  - **CurTot** - Current total.
  - **CurRate** - Current rate.
  - **MaxRate** - Maximum rate.
  - **AvgRate** - Average rate.
  - **PerTrans** - Per-transaction rate.

Selecting the "Logical Area Information" menu will now display two options: "Logical Area Overview (type)" and the previously existing "Logical Area Statistics".

The "system" logical areas can be filtered from the "Logical Area Overview" screen by selecting the "Display application logical areas" option of the "Tools" menu (obtained using the "!" shortcut). System logical areas can be included on the screen by selecting the "Display all logical areas" option of the "Tools" menu.

The "Logical Area Overview" screen statistic type can be specified using the configuration variable **LOGICAL_OVERVIEW_STAT** with the following keywords **CUR_TOTAL**, **CUR_RATE**, **MAX_RATE**, **AVG_RATE** and **PER_TRANS**.

The "Logical Area Overview" screen logical area type can be specified using the configuration variable **LOGICAL_OVERVIEW_TYPE** with the following keywords **TABLE**, **BTREE**, **HASH** and **BLOB**.

The "Logical Area Overview" screen can be configured to display application logical areas only (no "system" logical areas) using the configuration variable **SYSTEM_LOGICAL_AREAS** with the keyword **FALSE**. Specifying the configuration variable with the keyword **TRUE**, the default, will display all logical areas, including "system" logical areas.

The "Logical Area Overview" screen information is not saved in the binary output and, therefore, the screen is not available during binary file replay.

The "Logical Area Overview" screen is not available if the **/NOLOGICAL_AREA** qualifier is specified.

The "Logical Area Overview" screen participates in the "Cluster Statistic Collection" facility.

The following screen configuration options are available using the "Config" onscreen-menu option:

- **Modify column #1**. This option allows you to choose a different statistic field for column number 1.
- **Modify column #2**. This option allows you to choose a different statistic field for column number 2.
- **Modify column #3**. This option allows you to choose a different statistic field for column number 3.
- **Modify column #4**. This option allows you to choose a different statistic field for column number 4.
- **Change logical area type**. This option allows you to choose a different logical area type to be displayed on the screen. Selecting a new logical area type will reset the statistic fields to the default fields for that logical area type.

  Logical area types are: table, B-tree index, hash index and blob.
- **Change statistic type**. This option allows you to choose a different statistic type to be displayed on the screen. The selected statistic type applies to all statistic fields on the screen.

  Statistic types are: current total, current rate, maximum rate, average rate and per-transaction rate.

When selecting statistic fields for the various columns, no validation is performed to eliminate duplicate selections. This means you can display the same statistic field in one or more columns at the same time, if you so desire.

The following is an example of the "Logical Area Overview" screen for B-tree index logical areas:

```
Node: ALPHA3 (1/1/16)   Oracle Rdb X7.0-00 Perf. Monitor 18-MAR-1998 15:10:40.79
Rate: 1.00 Second    Logical Area Overview (Btree Indexes)  Elapsed: 04:18:42.51
Page: 1 of 1    KODH$:[R_ANDERSON.WORK.STATS]MF_PERSONNEL.RDB;1     Mode: Online
--------------------------------------------------------------------------------
Logical.Area.Name... leaf fetches leaf inserti leaf removal    discarded CurTot
COLL_COLLEGE_CODE.RD            0            0            0            0
DEG_EMP_ID.RDB$SYSTE          103            0            5            0
DEPARTMENTS_INDEX.DE          100            0            0            0
EMP_EMPLOYEE_ID.RDB$            5            0            3            0
JH_EMPLOYEE_ID.RDB$S            1            0            3            0
SH_EMPLOYEE_ID.RDB$S          103            0            3            0


--------------------------------------------------------------------------------
Config Exit Help Menu >next_page <prev_page Options Pause Reset Set_rate Write
```

The following is an example of the "Logical Area Overview" screen for hash index logical areas:

```
Node: ALPHA3 (1/1/16)   Oracle Rdb X7.0-00 Perf. Monitor 18-MAR-1998 15:11:09.68
Rate: 1.00 Second     Logical Area Overview (Hash Indexes)  Elapsed: 04:19:11.40
Page: 1 of 1    KODH$:[R_ANDERSON.WORK.STATS]MF_PERSONNEL.RDB;1     Mode: Online
--------------------------------------------------------------------------------
Logical.Area.Name... hash index f hash inserti hash deletio    discarded CurTot
EMPLOYEES_HASH.EMPID           37            0           37            0
EMPLOYEES_HASH.EMPID           57            0           57            0
EMPLOYEES_HASH.EMPID            6            0            6            0
JOB_HISTORY_HASH.EMP          235            0           37            0
JOB_HISTORY_HASH.EMP          343            0           57            0
JOB_HISTORY_HASH.EMP           50            0            6            0

--------------------------------------------------------------------------------
Config Exit Help Menu >next_page <prev_page Options Pause Reset Set_rate Write
```

The following is an example of the "Logical Area Overview" screen for blob logical areas:

```
Node: ALPHA3 (1/1/16)   Oracle Rdb X7.0-00 Perf. Monitor 18-MAR-1998 15:11:38.15
Rate: 1.00 Second        Logical Area Overview (Blobs)      Elapsed: 04:19:39.87
Page: 1 of 1    KODH$:[R_ANDERSON.WORK.STATS]MF_PERSONNEL.RDB;1     Mode: Online
--------------------------------------------------------------------------------
Logical.Area.Name... blob fetched  blob stored  blob erased    discarded CurTot
RDB$SEGMENTED_STRING           73            0            0            0


--------------------------------------------------------------------------------
Config Exit Help Menu >next_page <prev_page Options Pause Reset Set_rate Write
```

The "Logical Area Information" screen can also be sorted alphabetically and the user can "zoom in" on any displayed logical area to display that area's actual statistic information.

9.3.9 RMU/SHOW STATISTICS "Summary Tx Statistics" Screen

The **RMU/SHOW STATISTICS** utility has been enhanced to provide a "Summary Tx Statistics" screen. This screen summarizes database transaction activity and indicates transaction and verb execution rates.

The information displayed on the screen is a summation of the information displayed on a per-storage-area basis in the "IO Statistics" screens. This screen resides in the "Main" menu.

The following is an example of this screen:

```
Node: ALPHA3 (1/3/24)   Oracle Rdb X7.1-00 Perf. Monitor 10-JUL-1998 10:09:22.31
Rate: 1.00 Second            Summary Tx Statistics          Elapsed: 01:13:09.40
Page: 1 of 1       KODA_TEST:[R_ANDERSON.TCS_MASTER]TCS.RDB;2       Mode: Online
------------------------------------------------------------------------------
statistic.........      rate.per.second.............. total....... average......
name..............      max..... cur..... avg....... count....... per.trans....
transactions                   4        0        0.4         2065           1.0
  committed                    4        0        0.4         2065           1.0
  rolled back                  0        0        0.0            0           0.0
    duration x100              0        0        0.0            0           0.0
  prepared                     0        0        0.0            0           0.0

verb successes               455        6       27.1       119362          57.8
verb failures                  0        0        0.0            0           0.0
  duration x100                0        0        0.0            0           0.0

checkpoints                    1        0        0.0           42           0.0
  duration x100           133552        0       61.4       269839         130.6

RUJ file reads                 0        0        0.0            0           0.0
    file writes                4        1        0.5         2516           1.1
    file extend                1        0        0.0          313           0.1
------------------------------------------------------------------------------
Exit Graph Help Menu Options Pause Reset Set_rate Time_plot Write X_plot Yank !
```

Table 9–2 Screen Fields

Field	Description
transactions	The number of completed database transactions. This is the count of the COMMIT and ROLLBACK statements that have executed.
committed	The number of transactions that successfully updated the database.
rolled back	The number of transactions that were aborted and were not applied to the database.
duration x100	The duration of a transaction rollback operation, expressed in hundredths of a second displayed as a whole number. For example, the value "500" is "5" seconds.
prepared	The number of distributed transactions that have successfully "prepared" themselves for subsequent transaction commit.

(continued on next page)

Table 9–2 (Cont.) Screen Fields

Field	Description
verb successes	The number of completed verbs that returned a success status code. A verb is an atomic SQL statement or action. For example, a record insert and a record deletion are verbs. Also, within a compound statement each individual statement is atomic and Oracle Rdb performs a verb-success operation after processing each one. To avoid this overhead, you can use the SQL BEGIN ATOMIC statement to treat the entire block as a single verb.
verb failures	Give the number of completed verbs that returned an error status code. Errors include end-of-collection and deadlocks, as well as other exception conditions.
duration x100	Identifies the duration of a verb failure rollback operation, expressed in hundredths of a second displayed as a whole number. For example, the value "500" is "5" seconds.
checkpoints	Identifies the number of checkpoints performed by users. This field does not include the initial checkpoint when the user first attaches to the database.
duration x100	Displays the checkpoint duration, expressed in hundredths of a second displayed as a whole number. For example, the value "500" is "5" seconds.
RUJ file reads	Displays the total number of read I/O operations performed on the RUJ journal during the transaction undo phase. The RUJ file is never written by the database recovery (DBR) process. This field includes both synchronous and asynchronous I/O read requests.
file writes	Displays the total number of write I/O operations performed on the RUJ journal during the transaction phase. This field includes both synchronous and asynchronous I/O read requests.
file extends	Identifies the number of times an RUJ file has been extended.

9.3.10 RMU/SHOW STATISTICS "Recovery Information" Screen

This screen provides run-time standby database recovery information. It is important for analyzing network bandwidth utilization and standby database resource allocation effectiveness

This screen is only available on the standby database while Hot Standby is active. It resides in the "Hot Standby Information" menu.

The following is an example of the "Recovery Information" screen:

```
Node: BONZAI (1/1/24)   Oracle Rdb X7.1-00 Perf. Monitor 10-JUL-1998 10:09:57.37
Rate: 1.00 Second            Recovery Information          Elapsed: 01:16:51.59
Page: 1 of 1      KODA_TEST:[R_ANDERSON.TCS_STANDBY]TCS.RDB;2       Mode: Online
------------------------------------------------------------------------------
statistic.........      rate.per.second.............. total....... average......
name..............      max..... cur..... avg....... count....... per.trans....

transactions                  33        0        0.4         1979           1.0
 commit                       33        0        0.4         1979           1.0
 rollback                      0        0        0.0            0           0.0
 prepared                      0        0        0.0            0           0.0

Area ready                     0        0        0.0           12           0.0
```

```
AIJ records                3030        0        86.6        399769         202.0
 erase mixed                  0        0         0.0             0           0.0
 erase uniform                0        0         0.0             0           0.0
 modify mixed              1876        0        16.3         75399          38.0
 modify uniform            3030        0        70.3        324370         163.9

SPAM updated                806        0        13.8         63716          32.1

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

Table 9–3 Screen Fields

Field	Description
transactions	Gives the number of completed database transactions. This is the count of the COMMIT and ROLLBACK statements that have executed.
commit	Displays the number of transactions that have been committed to the standby database.
rollback	Displays the number of transactions that have been rolled back prior to being applied to the standby database.
prepared	Displays the number of distributed transactions that have been successfully "prepared" in anticipation of eventually being committed to the standby database.
Area ready	Displays the number of physical storage areas that have been "readied" during the recovery operation.
AIJ records	Displays the number of AIJ records applied.
erase mixed	Displays the number of "erase record" operations performed on a mixed-format storage area.
erase uniform	Displays the number of "erase record" operations performed on a uniform-format storage area.
modify mixed	Displays the number of "modify record" operations performed on a mixed-format storage area.
modify uniform	Displays the number of "modify record"operations performed on a uniform-format storage area.
SPAM updated	Displays the number of SPAM page modifications that occurred as a result of the AIJ journal record. SPAM pages are typically modified due to a live data page changing its threshold information.