Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Oracle® Rdb for OpenVMS

Release Notes

Release 7.0.6.4

June 2002

Oracle Rdb Release Notes, Release 7.0.6.4 for OpenVMS

Contents

Preface

Purpose of This Manual

This manual contains release notes for Oracle Rdb Release 7.0.6.4. 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 Rdb for OpenVMS Alpha and Oracle Rdb for OpenVMS VAX, which are referred to by their abbreviated name, Oracle Rdb.

Intended Audience

This manual is intended for use by all Oracle Rdb users. Read this manual before you install, upgrade, or use Oracle Rdb Release 7.0.6.4.

Document Structure

This manual consists of nine chapters:

Chapter 1	Describes how to install Oracle Rdb Release 7.0.6.4.
Chapter 2	Describes software errors corrected in Oracle Rdb Release 7.0.6.4.
Chapter 3	Describes software errors corrected in Oracle Rdb Release 7.0.6.3.
Chapter 4	Describes software errors corrected in Oracle Rdb Release 7.0.6.2.
Chapter 5	Describes software errors corrected in Oracle Rdb Release 7.0.6.1.
Chapter 6	Describes enhancements introduced in Oracle Rdb Release 7.0.6.4.
Chapter 7	Oracle Rdb Continuous LogMiner Documentation
Chapter 8	Provides information not currently available in the Oracle Rdb documentation set.
Chapter 9	Describes problems, restrictions, and workarounds known to exist in Oracle Rdb Release 7.0.6.4.

Chapter 1
Installing Oracle Rdb Release 7.0.6.4

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

NOTE

Beginning with Release 7.0.6.2 of Oracle Rdb, all new Oracle Rdb kits released are full kits. Oracle will no longer ship partial kits (known as ECOs in the past). Therefore, there is no need to install any prior release of Oracle Rdb when installing new Rdb kits.

1.1 Requirements

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

- Oracle Rdb 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 Rdb on all nodes in the cluster before proceeding.
- The installation requires approximately 100,000 free blocks on your system disk for OpenVMS VAX systems; 200,000 blocks for OpenVMS Alpha systems.

1.2 Invoking VMSINSTAL

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

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

variant−name

The variant names for the software update for Oracle Rdb Release 7.0.6.4 are:

- RDBSE4F070 for Oracle Rdb for OpenVMS VAX standard version.
- RDBASE4F070 for Oracle Rdb for OpenVMS Alpha standard version.
- RDBMVE4F070 for Oracle Rdb for OpenVMS VAX multiversion.
- RDBAMVE4F070 for Oracle Rdb 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:[RDBSE4F070.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 Alpha standard kit on device MTA0: and print the release notes:

```
$ @SYS$UPDATE:VMSINSTAL RDBASE4F070 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 Rdb installed will probably not be usable.

1.4 After Installing Oracle Rdb

This update provides a new Oracle Rdb Oracle TRACE facility definition. Any Oracle TRACE selections that reference Oracle Rdb will need to be redefined to reflect the new facility version number for the updated Oracle Rdb facility definition, "RDBVMSV7.0−64".

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

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

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

   ```
   $ LIBRARY /TEXT /EXTRACT=RDBVMSV7.0−64 −
   _$ /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 Rdb, the process executing the INSERT DEFINITION command must use the version of Oracle Rdb that matches the version used to create the Oracle TRACE administration database or the INSERT DEFINITION command will fail.

1.5 Maximum OpenVMS Version Check Added

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

OpenVMS Version 7.3 is the maximum supported version of OpenVMS.

As of Oracle Rdb Release 7.0.3, the Alpha EV6 processor is supported. As of Oracle Rdb Release 7.0.5, the Alpha EV67 processor is supported. As of Oracle Rdb Release 7.0.6, the Alpha Wildfire processor is supported (see http://metalink.oracle.com for specifics on which Wildfire configurations are supported). As of Oracle Rdb Release 7.0.6.2, the Alpha EV68 processor is supported.

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

Chapter 2
Software Errors Fixed in Oracle Rdb Release 7.0.6.4

This chapter describes software errors that are fixed by Oracle Rdb Release 7.0.6.4.

2.1 Software Errors Fixed That Apply to All Interfaces

2.1.1 Bugchecks Truncating Table in Mixed−Format Area With Row Caches

Bug 1994856

In some cases, truncating a table whose data or indexes are stored in mixed format areas can result in a bugcheck. This bugcheck was caused by incorrectly processing "Row Cache Reserved" space on a database page.

This problem has been corrected in Oracle Rdb Release 7.0.6.4. The "Row Cache Reserved" space on the database page now causes the row in the cache to be correctly fetched and considered for deletion.

2.1.2 Page Locks Not Always Demoted When Blocking AST Delivered

Bug 2245524

When global buffers were enabled, and asynchronous prefetch (APF) reading occurred, it was possible for a process to not release page locks when another process attempted to obtain the lock. That is, the blocking AST was ignored by the process that was holding the lock.

The RMU /SHOW LOCK command may show something similar to the following when this problem is encountered:

```
Resource: page 269
          ProcessID Process Name        Lock ID   System ID Requested Granted
          --------- ---------------     --------- --------- --------- -------
Waiting:  00000CA5  RVASIG01.......     6601649D  00000000  PW        CR
Blocker:  00000979  ACMS009SP009003     2600DABC  00000000            PR
```

In the above example, process 979 was holding the page lock in PR mode and process CA5 was attempting to promote its lock to PW mode. Process 979 never demoted the lock.

This problem was introduced in Oracle Rdb Release 7.0.6.2. Changes made in that release exposed a problem in the page lock blocking AST code.

This problem has been corrected in Oracle Rdb Release 7.0.6.4.

2.1.3 LRS Looping When Global Buffers Enabled

Bug 2061266

It was possible for the Log Recovery Server (LRS) to get into a CPU loop if the standby database had global buffers enabled and there was a high volume of updates being processed. This problem was introduced in Oracle Rdb Release 7.0.6.2 and was also in 7.0.6.3.

This problem has been corrected in Oracle Rdb Release 7.0.6.4.

2.1.4 Unresolved 2PC Transactions Rolled Back by RMU /RECOVER

Bug 2278911

When an RMU /RECOVER process completed processing the last journal specified, if the database was involved in a two−phase commit (2PC) transaction and the transaction was prepared but not yet committed (an "unresolved" transaction) when journal processing was complete, RMU /RECOVER would sometimes rollback the prepared transaction. Also, the "Current roll−forward sequence number" would be advanced to the next journal even though a transaction from the current journal was not completed.

This behavior was incorrect since unresolved transactions should be considered still active and must remain active until a commit or rollback record is found in the journal or the user explicitly instructs RMU /RECOVER to commit or abort the 2PC transaction. Advancing the "Current roll−forward sequence number" also allowed subsequent RMU /RECOVER commands to not require the journal(s) that contained the unresolved transaction. If the journal(s) containing the unresolved transaction was not applied again, the unresolved transaction would be lost.

When this situation occurred, output similar to the following would be observed from the RMU /RECOVER command:

```
%RMU−I−AIJACTIVE, 1 active transaction not yet committed or aborted
%RMU−I−LOGRECSTAT, transaction with TSN 0:143 is active
%RMU−I−AIJPREPARE, 1 of the active transactions prepared but not yet committed
or aborted
%RMU−I−AIJSUCCES, database recovery completed successfully
%RMU−I−AIJNXTSEQ, to continue this AIJ file recovery, the sequence number
needed will be 1
%RMU−I−LOGRECSTAT, transaction with TSN 0:143 rolled back
  .
  .
  .
%RMU−I−AIJFNLSEQ, to start another AIJ file recovery, the sequence number
needed will be 1
```

Note that in this example the active transaction was rolled back even though it was not yet resolved. Also, the sequence number was advanced to the next journal even though the active transaction had not been resolved.

One situation where this could occur is when the prepare record was stored in one journal but the commit record was stored in the next journal. In that situation, the transaction could be lost if multiple RMU /RECOVER commands were used to recover the database. To prevent that from occurring, all available journals should be specified in a single RMU /RECOVER command. That is, there should not be a separate RMU /RECOVER command issued for each journal; all journals must be applied by a single RMU /RECOVER command.

This problem has been corrected in Oracle Rdb Release 7.0.6.4. At the end of journal processing, if there is an unresolved transaction still active, the transaction will remain active and the "Current roll−forward sequence number" will not be advanced.

2.1.5 Bugchecks at PSII2SCANSTARTBBCSCAN

In prior releases of Oracle Rdb, it was possible that a query involving SORTED RANKED indexes could bugcheck when trying to establish a scan of a duplicate node.

```
***** Exception at 00A2EA30 : PSII2SCANSTARTBBCSCAN + 000004F8
```

```
%COSI-F-BUGCHECK, internal consistency failure
```

This condition only occurs with SORTED RANKED indexes where a sequence of inserts, updates and deletes of the same duplicate values force the production of an overflow duplicates node but subsequent deletes remove the duplicate entries that are on the primary index node for that duplicate value.

A possible workaround for this problem is to rebuild the sorted ranked index.

This problem has been corrected in Oracle Rdb Release 7.0.6.4.

2.1.6 Cursor on Ranked Index Returned too Many Records

Bug 2270786

A problem in the way the current record offset was determined for SORTED RANKED index duplicate entries may cause Oracle Rdb to return the same record twice on a table cursor fetch.

This problem would only occur given the following circumstances:

- A cursor is established on a table and strategy shows that a sorted ranked index will be used to retrieve the records.
- The cursor fetch returned the first duplicate record in a duplicate entry with exactly two duplicates.
- The same process with this cursor open inserts a new record into or removes another record from the same table.
- The insert or delete happened to update the same index node currently referenced by the cursor.

In this situation, Oracle Rdb must invalidate the current fetch scan and re−establish its currency. However, the currency was incorrectly set to the first duplicate in the current entry, hence returning this record a second time on the next fetch.

Workarounds for this problem include:

- Rebuilding the index may provide a temporary workaround for this problem.
- Change the processing of the records so as to not interleave fetches and inserts in the same process in this manner.
- Alternatively, rebuild the index as a normal SORTED index.

This problem has been corrected in Oracle Rdb Release 7.0.6.4.

2.1.7 Bugcheck at RDMS$$ALPHA$CONVERT_SORT+00000778

Bug 2220891

Under certain conditions, a query would result in a bugcheck in routine RDMS$$ALPHA$CONVERT_SORT or RDMS$$CONVERT_SORT.

The following is an example which would show the problem. Create two tables with similar columns. One set of columns are similar but not the same: INT and SMALLINT. The other set of columns are the same type, VARCHAR.

```
create table table1 (column1 smallint, column2 varchar(5));
create table table2 (column1 int,      column2 varchar(5));

insert into table1 values (100, 'abcde');
```

```
insert into table2 values (100, 'abcde');

create index table1_idx on table1 (column1, column2);
create index table2_idx on table2 (column1, column2);

select t1.column1, t2.column2 from table1 t1, table2 t2
  where t1.column1 = t2.column1 and t1.column2 = t2.column2;
```

The bugcheck error would occur during compilation of the select statement.

This problem can be avoided by first disabling zigzag match strategies before executing this particular SELECT statement. To disable zigzag match, SET FLAGS 'NOZIGZAG_MATCH' in interactive or dynamic SQL. Then execute the SELECT statement and re−enable zigzag match for subsequent queries. If you do not have that degree of control, then $DEFINE RDMS$SET_FLAGS NOZIGZAG_MATCH prior to executing the application program. Doing so will disable zigzag match strategies for all queries executed by that application.

This problem has been corrected in Oracle Rdb Release 7.0.6.4.

2.1.8 Privileged User Bugcheck (ACCVIO)

Bug 2297264

A privileged user with no access granted to the database could receive an ACCVIO error and a bugcheck when executing actions outside of a transaction (for example, calling a stored procedure).

The following example shows the bugcheck exception report:

```
***** Exception at 00000004 : symbol not found
%SYSTEM−F−ACCVIO, access violation, reason mask=00, virtual
    address=0000000000000004, PC=0000000000000004, PS=0000000B
```

A possible workaround is to grant access to the user.

This problem has been corrected in Oracle Rdb Release 7.0.6.4.

2.1.9 RDMS$CREATE_LAREA_NOLOGGING Partly Ignored for Objects with Row Caches

Bug 2155224

When using the RDMS$CREATE_LAREA_NOLOGGING logical name to avoid after−image journaling when creating database objects that were cached (indexes, for example), it was possible that the after−image journal was still being written to for each modified row. This resulted in unexpected journal growth.

This problem has been corrected in Oracle Rdb Release 7.0.6.4. The RDMS$CREATE_LAREA_NOLOGGING logical name setting now correctly avoids writing to the after−image journal for cached objects.

2.1.10 Exception in RDMS$$KOD_ISCAN_GET_NEXT

Bug 2225971

An ACCVIO exception and bugcheck could occur when range−list processing was in use.

An example of a query that would cause this error follows.

```
SQL> select a.case_id, b.pos_nr, b.h_line_num
cont> from
cont> case a, case_line b
cont> where
cont> a.case_num = b.case_num and
cont> a.case_status > 30 and
cont> (b.rec_num = '' or b.rec_num is null) and
cont> exists (select '*' from n_rec_line c
cont>         where b.h_line_num=c.h_line_num and
cont>           (c.rec_num = '' or c.rec_num is null) and
cont>           (not exists (select '*' from c_next_line d
cont>                        where d.line_num = c.line_num) and
cont>                         ((c.coll_num = '' or c.coll_num is null) or
cont>                         (c.coll_num <>  '' and c.coll_num is not null and
cont>                          not exists (select '*' from p_coll d
cont>                                      where d.coll_num = c.coll_num)
cont>                    )
cont>           )
cont>           )
cont>    )
cont> order by a.case_id, b.pos_nr;
Sort                       Conjunct
Match
  Outer loop
    Sort
    Cross block of 2 entries
      Cross block entry 1
        Conjunct                  Get
        Retrieval sequentially of relation CASE_LINE
      Cross block entry 2
        Conjunct                  Get
        Retrieval by index of relation CASE
          Index name  CA00_HIDX_PS [1:1]                    Direct lookup
  Inner loop
    Aggregate                 Sort
    Cross block of 3 entries
      Cross block entry 1
        Leaf#01 BgrOnly N_REC_LINE Card=48527
          BgrNdx1 RE20_HIDX_S [(1:1)2] Fan=1
          BgrNdx2 RE20_COLLI_NUM_SIDX [0:1,(1:1)2] Bool Fan=44
      Cross block entry 2
        Conjunct                  Aggregate-F1
        Index only retrieval of relation P_COLL
          Index name  PM10_HIDX_PS [1:1]
      Cross block entry 3
        Conjunct                  Aggregate-F1
        Index only retrieval of relation C_NEXT_LINE
          Index name  CF20_SIDX_P [1:1]
%RDMS-I-BUGCHKDMP, generating bugcheck dump file DISK1:[TEST]RDSBUGCHK.DMP;
```

The exception report in the bugcheck dump file is:

```
***** Exception at 0147DC64 : RDMS$$KOD_ISCAN_GET_NEXT + 00001804
%SYSTEM-F-ACCVIO, access violation, reason mask=00,
virtual address=000000004F435F50, PC=000000000147DC64, PS=00000009
```

This problem has been corrected in Oracle Rdb Release 7.0.6.4.

2.1.11 Records Incorrectly Applied to a Key Entry with Sorted Ranked Index

Bug 2322296

A problem in the way index entry currency was determined caused incorrect assignment of record identifiers to index entries resulting in wrong results when the index was used in a search.

This problem only occurs with Sorted Ranked Indexes.

RMU /VERIFY of the index will show this problem as an informational or error message stating that the cardinalities are inconsistent.

```
%RMU-I-BTRDUPCAR, Inconsistent duplicate cardinality (C1) of 224 specified
                  for entry 1 at dbkey 60:52:1.
                  Actual count of duplicates is 2.
%RMU-I-BTRDUPCAR, Inconsistent duplicate cardinality (C1) of 36 specified
                  for entry 2 at dbkey 60:52:1.
                  Actual count of  duplicates is 2.
%RMU-I-BTRERPATH, parent B-tree node of 60:52:1 is at 60:50:0
%RMU-I-BTRDUPCAR, Inconsistent duplicate cardinality (C1) of 130 specified
                  for entry 1 at dbkey 60:53:1.
                  Actual count of duplicates is 386.
%RMU-I-BTRERPATH, parent B-tree node of 60:53:1 is at 60:50:0
%RMU-I-BTRROODBK, root dbkey of B-tree is 60:50:0
%RMU-I-NDXERRORS,  3 index errors encountered
```

It is highly recommended that sorted ranked indexes be verified regularly using RMU /VERIFY to determine if this problem has occurred.

A possible workaround for this problem is to rebuild the affected indexes as this problem does not occur during an index build.

In addition, affected indexes should be rebuilt after upgrading to Oracle Rdb Release 7.0.6.4 as this problem does affect the index data stored on-disk.

This problem has been corrected in Oracle Rdb Release 7.0.6.4.

2.1.12 Query With OR and Repeated AND Predicates Looped Forever

Bugs 2332845 & 370844

A simple SQL query with redundant terms in its WHERE clause would never finish. It would remain in a never-ending CPU loop until the process executing the query was stopped.

The following query on the PERSONNEL database shows the problem:

```
select last_name, middle_initial from employees
  where ((middle_initial = 'I' and state = 'NH') and
         (middle_initial = 'O' and state = 'NH'))
         or
         (middle_initial = 'T' and state = 'NH');
```

The first leg of the OR expression contains four ANDed terms, two of which are the same (state = 'NH').

There exists logic in the Rdb optimizer to consolidate common expressions. In this case, that would apply to the multiple instances of the state = 'NH' predicate. That logic was faulty and could result in the query never completing with the optimizer looping forever trying to process these expressions.

This problem can be avoided by rewriting the query such that the common term, state = 'NH', is factored out and used only once. It is also true that in this particular case, the first part of the OR expression can be removed completely because the results must always evaluate to false (middle_initial = 'I' and middle_initial = 'O'). That happens to be the form of the query submitted in a customer bug report.

This problem has been corrected in Oracle Rdb Release 7.0.6.4.

2.1.13 Query With Same Column in Two Clauses Returns Wrong Results

Bug 2285818

The following query with the same column in two clauses should return one row.

```
set flags 'strategy,detail';

SELECT T2.vert, T3.flag, T5.data
        FROM T1, T2, T3, T4, T5
        WHERE T1.plan_id = T2.plan_id
           AND T1.cust_id = T4.cust_id
           AND T3.prod_id = T4.prod_id
           AND T5.prod_id = T4.prod_id
           AND T3.prod_id = T5.prod_id
           AND T1.code = ' '
           AND ((T4.prio = 3 AND T3.flag = '10')   <== "T3.flag = '10'" is
             OR (T4.prio = 3 AND T3.flag = '12')
             OR (T4.prio = 3 AND T3.flag = '13'))
           AND ((T3.flag = '10' AND T5.data = '73' ) <== reused here again
                 OR T5.data <> '73' ) ;
Tables:
  0 = T1
  1 = T2
  2 = T3
  3 = T4
  4 = T5
Cross block of 5 entries
  Cross block entry 1
    Conjunct: 0.CODE = ' '
    Get     Retrieval sequentially of relation 0:T1
  Cross block entry 2
    Get     Retrieval by index of relation 1:T2
      Index name  I_T2_01 [1:1]  Direct lookup
        Keys: 0.PLAN_ID = 1.PLAN_ID
  Cross block entry 3
    Conjunct: 3.PRIO = 3
    Get     Retrieval by index of relation 3:T4
      Index name  I_T4_01 [1:1]         Direct lookup
        Keys: 0.CUST_ID = 3.CUST_ID
  Cross block entry 4
    Conjunct: (2.FLAG = '10') OR (2.FLAG = '12') OR
              (2.FLAG = '13')
    Get     Retrieval by index of relation 2:T3
      Index name  I_T3_01 [1:1]        Direct lookup
        Keys: 2.PROD_ID = 3.PROD_ID
  Cross block entry 5
    Conjunct: (2.PROD_ID = 4.PROD_ID) AND
```

```
            ((4.DATA = '73') OR (4.DATA <> '73')) <== missing "FLAG = '10'"
    Get     Retrieval by index of relation 4:T5
      Index name  I_T5_01 [1:1]        Direct lookup
        Keys: 4.PROD_ID = 3.PROD_ID
 T2.VERT       T3.FLAG  T5.DATA
 LV_508        13       73                    <== WRONG
 LH_610        12       75                    <== CORRECT
2 rows selected
```

One of the equality predicates in the OR clauses referencing table T3 is referenced again in another clause, as shown below.

```
    AND ((T4.prio = 3 AND T3.flag = '10')   <== "T3.flag = '10'" is
      OR (T4.prio = 3 AND T3.flag = '12')
      OR (T4.prio = 3 AND T3.flag = '13'))
    AND ((T3.flag = '10' AND T5.data = '73' ) <== reused here again
          OR T5.data <> '73' ) ;
```

However, in the detailed strategy display, the predicate is missing under the cross block entry 5, as shown below.

```
  Cross block entry 5
    Conjunct: (2.PROD_ID = 4.PROD_ID) AND
              ((4.DATA = '73') OR (4.DATA <> '73')) <== missing "FLAG = '10'"
    Get     Retrieval by index of relation 4:T5
      Index name  I_T5_01 [1:1]        Direct lookup
        Keys: 4.PROD_ID = 3.PROD_ID
```

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

1. The main query joins 5 tables (T1, T2, T3, T4, T5) using cross strategy with 5 cross block entries. Tables T3, T4 and T5 are joined by PROD_ID column key, table T4 is joined with T1 by CUST_ID, and T1 is joined with T2 by PLAN_ID.
2. Table T1 rows are filtered by an equality predicate. If this is removed, the strategy changes in the order of the cross blocks and the query works.
3. One of the filter predicates contain OR expressions which reference one column of table T4 (T4.PRIO) and one column of table T3 (T3.FLAG).
4. Another filter predicate contains an OR expression which references the same column of table T3 from the previous filter predicate (e.g. T3.FLAG = '10'). This is the main reason why the query returns wrong results.

As a workaround, the query works if the second predicate "T3.FLAG = '10'" is replaced by a LIKE operator, such as "T3.FLAG like '10'".

```
set flags 'strategy,detail';

SELECT T2.vert, T3.flag, T5.data
        FROM T1, T2, T3, T4, T5
        WHERE T1.plan_id = T2.plan_id
           AND T1.cust_id = T4.cust_id
           AND T3.prod_id = T4.prod_id
           AND T5.prod_id = T4.prod_id
           AND T3.prod_id = T5.prod_id
           AND T1.code = ' '
           AND ((T4.prio = 3 AND T3.flag = '10')   <== "T3.flag = '10'" is
             OR (T4.prio = 3 AND T3.flag = '12')
             OR (T4.prio = 3 AND T3.flag = '13'))
           AND ((T3.flag LIKE '10' AND T5.data = '73' ) <== replaced by LIKE
                 OR T5.data <> '73' ) ;
```

This problem has been corrected in Oracle Rdb Release 7.0.6.4.

2.1.14 GROUP BY Query Followed by CASE With EXISTS Clause Returns Wrong Results

Bug 2198990

The following GROUP BY query followed by CASE with EXISTS should return 3 rows but returns only 2 rows.

```
set flags 'strategy,detail';

select count(*), RD.York_Loss_Code,
    CASE WHEN EXISTS (Select * from Loss_Gruppe where
                         Loss_Code = RD.York_Loss_Code
                        )
         THEN 'P'
         ELSE 'F' END
    from redraw RD
group by RD.York_Loss_Code,
    CASE WHEN EXISTS (Select * from Loss_Gruppe where
                         Loss_Code = RD.York_Loss_Code
                        )
         THEN 'P'
         ELSE 'F' END
optimize using test_outline
;
~S: Outline "TEST_OUTLINE" used
Aggregate       Sort
Match
  Outer loop
    Sort    Get     Retrieval sequentially of relation REDRAW
  Inner loop
    Aggregate       Sort    Get
    Retrieval sequentially of relation LOSS_GRUPPE
                YORK_LOSS_CODE
            1   1               P
            1   2               P
2 rows selected

where the tables contain the following rows:

select york_loss_code from redraw;
 YORK_LOSS_CODE
 1
 10
 2
3 rows selected

select loss_code from loss_gruppe;
 LOSS_CODE
 1
 2
2 rows selected
```

This feature was not included in the very first release of Oracle Rdb 7.0 and this is the first time the customer has used the GROUP BY clause followed by the CASE with EXISTS clause.

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

1. The main select query is a count aggregate with GROUP BY clause.
2. One of columns in the GROUP BY clause contains a CASE expression with an EXISTS clause on a subquery.

As a workaround, the query works if the query outline TEST_OUTLINE is changed to use cross strategy, as in the following example.

```
create outline TEST_OUTLINE
id '1B91E858006B77EC167036406D2D04AB'
mode 0
as (
  query (
    subquery (
      subquery (
        REDRAW 0        access path sequential
          join by cross to
!         join by match to
        subquery (
          LOSS_GRUPPE 1   access path sequential
          )
        )
      )
    )
  )
compliance optional ;
```

This problem has been corrected in Oracle Rdb Release 7.0.6.4.

2.1.15 Query With Join Predicates on Leading Segments and Equality Filters Returns Wrong Results

Bug 2204152

The following query with join predicates on leading segments and equality filters should find 2 rows instead of 0 rows.

```
set flags 'strategy,detail';
SELECT T2.PRICE_AMT FROM T1, T2
  WHERE
    T2.CMP_NO = 1 AND
    T2.PROD_NO = 161255 AND
    T2.DIV_NO = 1 AND
    T2.CUST_NO = 10674 AND

    T1.CMP_NO = T2.CMP_NO AND
    T1.PROD_NO = T2.PROD_NO AND
    T1.DIV_NO = T2.DIV_NO AND
    T1.CUST_NO = T2.CUST_NO AND
    T1.QUOTE = 0
    ;
Tables:
  0 = T1
  1 = T2
Cross block of 2 entries
  Cross block entry 1
    Conjunct: 0.QUOTE = 0
    Conjunct: 0.DIV_NO = 1
    Conjunct: 0.CMP_NO = 1
    Index only retrieval of relation 0:T1
      Index name  T1_NDX [4:4]
```

```
        Keys: (0.DIV_NO = 1) AND (0.PROD_NO = 161255) AND
              (0.CUST_NO = 10674) AND
              (0.CMP_NO = 1.CMP_NO)  <== Note1: incorrect conjunct
  Cross block entry 2
    Leaf#01 FFirst 1:T2 Card=7843
      Bool: (1.CMP_NO = 1) AND (1.PROD_NO = 161255) AND (1.DIV_NO =
            1) AND (1.CUST_NO = 10674) AND (0.CMP_NO = 1.CMP_NO)
             AND (0.PROD_NO = 1.PROD_NO) AND (0.DIV_NO =
            1.DIV_NO) AND (0.CUST_NO = 1.CUST_NO)
      BgrNdx1 T2_NDX [2:2] Fan=13
        Keys: (0.CUST_NO = 1.CUST_NO) AND (0.PROD_NO = 1.PROD_NO)
        Bool: (1.CMP_NO = 1) AND (1.PROD_NO = 161255) AND (1.DIV_NO
              = 1) AND (1.CUST_NO = 10674)
0 rows selected
```

Note1: 1.CMP_NO references table T2 in the cross block entry 1 where context 1 is not available yet.

```
Indexes on table T1:
T1_NDX                          with column CUST_NO
                                and column PROD_NO
                                and column DIV_NO
                                and column CMP_NO
                                and column QUOTE

Indexes on table T2:
T2_NDX                          with column CUST_NO
                                and column PROD_NO
                                and column START_DATE
                                and column DIV_NO
                                and column CMP_NO
```

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

1. The query joins two tables, T1 and T2, using all leading segments except the last one in T1_NDX index, e.g. T1.CMP_NO, T1.PROD_NO, T1.DIV_NO, T1.CUST_NO.
2. The last segment T1.QUOTE of T1_NDX is also used in the equality filter.
3. There is also an equality filter for each segment of T2_NDX used as join predicate, e.g. T2.CMP_NO, T2.PROD_NO, T2.DIV_NO, T2.CUST_NO.

As a workaround, the query works if the SQL flag TRANSITIVITY is turned off.

```
set flags 'notransitivity, max_stability';

Tables:
  0 = T1
  1 = T2
Cross block of 2 entries
  Cross block entry 1
    Get     Retrieval by index of relation 1:T2
      Index name  T2_NDX [2:2]
        Keys: (1.CUST_NO = 10674) AND (1.PROD_NO = 161255)
        Bool: (1.CMP_NO = 1) AND (1.DIV_NO = 1)
  Cross block entry 2
    Conjunct: 0.QUOTE = 0
    Index only retrieval of relation 0:T1
      Index name  T1_NDX [4:4]
        Keys: (0.DIV_NO = 1.DIV_NO) AND (0.PROD_NO = 1.PROD_NO)
              AND (0.CUST_NO = 1.CUST_NO) AND
              (0.CMP_NO = 1.CMP_NO)             <== Note2
   T2.PRICE_AMT
          29.12
```

```
          29.12
2 rows selected
```

Note2: 1.CMP_NO references table T2 in the cross block entry 2 where context 1 is already made available in the cross block entry 1.

It also works if the optimizer statistics are collected by running RMU /COLLECT on this table T1.

```
Tables:
  0 = T1
  1 = T2
Cross block of 2 entries
  Cross block entry 1
    Conjunct: 0.QUOTE = 0
    Conjunct: 0.DIV_NO = 1
    Conjunct: 0.CMP_NO = 1
    Index only retrieval of relation 0:T1
      Index name  T1_NDX [5:5]    Direct lookup
        Keys: (0.CMP_NO = 1) AND
              (0.DIV_NO = 1) AND (0.PROD_NO = 161255) AND
              (0.CUST_NO = 10674) AND (0.QUOTE = 0) <== Note3
  Cross block entry 2
    Conjunct: (0.CMP_NO = 1.CMP_NO) AND (0.DIV_NO = 1.DIV_NO)
    Get     Retrieval by index of relation 1:T2
      Index name  T2_NDX [2:2]
        Keys: (0.CUST_NO = 1.CUST_NO) AND (0.PROD_NO = 1.PROD_NO)
        Bool: (1.CMP_NO = 1) AND (1.DIV_NO = 1) AND (1.CUST_NO =
              10674) AND (1.PROD_NO = 161255)
  CQD.PRICE_AMT              CQD.COST_AMT   CQD.DEAL_AMT
          29.12                     27.85           1.50
          29.12                     27.85           1.50
2 rows selected
```

Note3: Only table context 0 is referenced in the cross block entry 1. No reference is made to context 1 of table T2.

This problem has been corrected in Oracle Rdb Release 7.0.6.4.

2.1.16 Query With Transitive Join Predicates and Non−equality Filter Bugchecks

Bug 2207963

The following query works in Oracle Rdb 7.0.6.2 but bugchecks in Oracle Rdb 7.0.6.3:

```
set flags 'strategy,detail';

select T1.PROC_CD, T1.SYS_CD,
       T2.RUN_NBR, T2.CALENDER, T2.PROC_COD
  From T1, T2, T3
  where     T2.SYS_CD    = T1.SYS_CD
        AND T2.PROC_CD   = T1.PROC_CD
        AND T2.SEQ_NBR   = T1.SEQ_NBR
        AND T2.CYCLE_CD  = T1.CYCLE_CD
        AND T2.PROFIL_CD = T1.PROFIL_CD

        AND T3.SYS_CD    = T2.SYS_CD
        AND T3.PROC_CD   = T2.PROC_CD
        AND T3.DAY_DATE  = T2.CALENDER
        AND T3.SEQ_NBR   = T2.SEQ_NBR
```

```
        AND T3.RUN_NBR   = T2.RUN_NBR
        AND T3.CYCLE_CD  = T2.CYCLE_CD
        AND T3.PROFIL_CD = T2.PROFIL_CD

        AND T1.SYS_CO = 'CPD'
        AND T2.CALENDER <= '15-JAN-2002'
   ;
```

Note: All the leading segments except the last one in T2_NDX index are used as join predicates.

```
Indexes on table T2:
T2_NDX                          with column SYS_CD
                                and column PROC_CD
                                and column CALENDER
                                and column SEQ_NBR
                                and column RUN_NBR
                                and column CYCLE_CD
                                and column PROFIL_CD
                                and column PROC_COD
```

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

1. The query joins 3 tables, T1, T2, T3 where table T1 and T3 are joined via transitive selection predicates such as "T1.col = T2.col and T2.col = T3.col".
2. Almost all of the leading segments except the last one in the index T2_NDX are referenced in the transitive predicates.
3. The filter predicate that references the 3rd leading segment, CALENDER, is a non−equality using "<=" operator.

As a workaround, the query works if the SQL flag TRANSITIVITY is turned off.

This problem has been corrected in Oracle Rdb Release 7.0.6.4.

2.1.17 Query With OR Predicates Including Two Similar IS NULL Clauses Returns Wrong Results

Bug 2177832

The following query with OR predicates including two similar IS NULL clauses should return 8 rows but instead returns 0 rows:

```
set flags 'strategy,detail';

select police.no_contra, police.cd_typol
from CMFasssoc asssoc, CMFpolice police, CMFassfam assfam, CMFserpol serpol
where
asssoc.statut <> 2
and asssoc.no_assure = 1670
and police.no_assure = asssoc.no_assure
and police.statut <> 2
and ((police.no_contra is null)
  or (police.no_contra is NOT null AND POLICE.CD_TYPOL <> 0)
    )
and assfam.no_assure = asssoc.no_assure
and assfam.statut <> 2
and serpol.no_assure = asssoc.no_assure
and serpol.no_police = police.no_police
and serpol.datd_mfac in (select max(serpol2.datd_mfac)
```

```
                        from CMFserpol serpol2
                        where serpol2.no_assure = asssoc.no_assure
                        and serpol2.no_police = police.no_police
                       )
;
Tables:
  0 = CMFASSSOC
  1 = CMFPOLICE
  2 = CMFADCLI2
  3 = CMFSERPOL
  4 = CMFSERPOL
Cross block of 4 entries
  Cross block entry 1
    Conjunct: 2.NO_CLI = 0.NO_ASSURE
    Match
      Outer loop
        Conjunct: 2.STATUT <> 2
        Conjunct: (2.TYP_CLI = 1) AND (2.CFCA_CLI2 = 1) AND (2.TYP_CLIRE = 2)
        Leaf#01 Sorted 2:CMFADCLI2 Card=2
          Bool: 2.NO_CLI = 1670
          FgrNdx  CMFADCLI2_I1 [3:3] Fan=9
            Keys: (2.NO_CLI = 1670) AND (2.TYP_CLI = 1) AND (2.CFCA_CLI2 = 1)
          BgrNdx1 CMFADCLI2_I2 [1:1] Fan=9
            Keys: 2.TYP_CLIRE = 2
            Bool: (2.NO_CLI = 1670) AND (2.TYP_CLI = 1) AND (2.CFCA_CLI2 = 1)
      Inner loop      (zig-zag)
        Conjunct: (0.STATUT <> 2) AND (0.NO_ASSURE = 1670)
        Get     Retrieval by index of relation 0:CMFASSSOC
          Index name  CMFASSSOC_I1 [1:1]
            Keys: 0.NO_ASSURE = 1670
  Cross block entry 2
    Conjunct: 2.STATUT <> 2
    Leaf#02 FFirst 1:CMFPOLICE Card=2
      Bool: (0.STATUT <> 2) AND (0.NO_ASSURE = 1670) AND (1.NO_ASSURE =
            0.NO_ASSURE) AND (2.NO_CLI = 0.NO_ASSURE)
      BgrNdx1 POLICE_H_IDX_1 [1:1] Fan=1
        Keys: 1.NO_ASSURE = 0.NO_ASSURE
        Bool: 1.NO_ASSURE = 1670
      BgrNdx2 CMFPOLICE_I2 [0:1,1:1] Fan=12
        Keys: r0: NOT MISSING (1.NO_CONTRA)
              r1: MISSING (1.NO_CONTRA)
  Cross block entry 3
    Aggregate: 0:MAX (4.DATD_MFAC)
    Conjunct: (1.STATUT <> 2) AND <error: missing expression> <== NOTE (1)
              AND (1.CD_TYPOL <> 0)
    Conjunct: 4.NO_POLICE = 1.NO_POLICE
    Get     Retrieval by index of relation 4:CMFSERPOL
      Index name  SERPOL_H_IDX_1 [1:1]
        Keys: 4.NO_ASSURE = 0.NO_ASSURE
        Bool: 4.NO_ASSURE = 1670
  Cross block entry 4
    Conjunct: (3.NO_POLICE = 1.NO_POLICE) AND (3.DATD_MFAC = <agg0>)
    Get     Retrieval by index of relation 3:CMFSERPOL
      Index name  SERPOL_H_IDX_1 [1:1]
        Keys: 3.NO_ASSURE = 0.NO_ASSURE
        Bool: 3.NO_ASSURE = 1670
0 rows selected

NOTE (1) : Error in the conjunct indicates some expression is missing.
           This is the cause of the problem.
```

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

1. The query joins 3 tables and one simple view.
2. The WHERE clause includes several join predicates, some filter predicates, and an IN clause on a subquery.
3. One of the filter predicates contains an OR expression with similar IS NULL clauses on each branch.

As a workaround, the query works if the SQL flag 'MAX_STABILITY' is set.

This problem has been corrected in Oracle Rdb Release 7.0.6.4.

2.1.18 UNION Query With Constant Column Returns Wrong Results

Bug 2231693

The following UNION query with constant column should return 1 row.

```
set flags 'strategy,detail';

create table t1 (art_no char(12), art_rev char(12));
create table t2  (art_no char(12), art_rev char(12));
insert into t1 values ('053 2021-120', ' ');
create view t1_view
  as select
  art_no, art_rev from t1;
create view t2_view
  as select
  art_no, art_rev from t2;

select v.art_no, v.art_rev
from (
      select adr.*, 'X' as RevType
      from t2_view adr
      union
      select a.*, ' ' as RevType
      from t1_view a
     ) v
where
    v.art_no = '053 2021-120' and
    RevType = ' ' ;

Tables:
  0 = T2
  1 = T1
Merge of 1 entries
  Merge block entry 1
  Reduce: <mapped field>, <mapped field>, <mapped field>
  Sort: <mapped field>(a), <mapped field>(a), <mapped field>(a)
  Conjunct: 'X' = ' '                   <== Note1
  Merge of 2 entries
    Merge block entry 1
    Conjunct: 0.ART_NO = '053 2021-120'
    Conjunct: 'X' = ' '
    Get     Retrieval sequentially of relation 0:T2
    Merge block entry 2
    Conjunct: 1.ART_NO = '053 2021-120'
    Conjunct: ' ' = ' '
    Get     Retrieval sequentially of relation 1:T1
0 rows selected
```

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

1. The query selects from the derived table of 2 unioned subselect queries which select all columns plus additional constant column from the simple view of each table.
2. The WHERE clause contains the equality predicate referencing the constant column of the unioned derived table.

There is no known workaround for this problem.

This problem has been corrected in Oracle Rdb Release 7.0.6.4.

2.1.19 Query With CAST Function Using Ranked Index Signals Exception Error

Bug 2235593

The following query with a CAST function using a ranked index signals an exception error:

```
create table t1 (y2k smallint, data_id char(11), proj_id char(6));
insert into t1 values (20,'20020202','2915');
create unique index t1_i1 on t1 (y2k, data_id) type is sorted ranked;
create unique index t1_i2 on t1 (proj_id) type is sorted;

select * from t1 where
        proj_id='2915' and
       cast (data_id as integer) = 20020202;
Leaf#01 FFirst 0:T1 Card=1
  Bool: (0.PROJ_ID = '2915') AND (CAST (0.DATA_ID AS INT) = 20020202)
  BgrNdx1 T1_I1 [0:0] Fan=12
    Bool: CAST (0.DATA_ID AS INT) = 20020202
  BgrNdx2 T1_I2 [1:1] Fan=16
    Keys: 0.PROJ_ID = '2915'
%RDB-E-ARITH_EXCEPT, truncation of a numeric value at runtime
-COSI-F-INPCONERR, input conversion error
```

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

1. The query contains a WHERE clause with 2 equality predicates. One of the predicates uses a CAST function.
2. The query uses dynamic optimizer strategy with 2 background indices where the first one is a ranked index.
3. The first background index has 2 segments where the second segment is referenced by the CAST function in the WHERE clause.

As a workaround, the query works if the dynamic optimizer is disabled by setting the SQL flag MAX_STABILITY. The query also works if T1_I1 is a non−ranked sorted index.

```
drop index t1_i1;
create unique index t1_i1 on t1 (y2k, data_id) type is sorted ;

select * from t1 where
        proj_id='2915' and
       cast (data_id as date vms) = '02-Feb-2002';
Leaf#01 FFirst 0:T1 Card=1
  Bool: (0.PROJ_ID = '2915') AND (CAST (0.DATA_ID AS DATE VMS) = '02-FEB-2002')
  BgrNdx1 T1_I1 [0:0] Fan=12
    Bool: CAST (0.DATA_ID AS DATE VMS) = '02-FEB-2002'
  BgrNdx2 T1_I2 [1:1] Fan=16
```

```
    Keys: 0.PROJ_ID = '2915'
    Y2K   DATA_ID       PROJ_ID
     20   20020202      2915
1 row selected
```

This problem has been corrected in Oracle Rdb Release 7.0.6.4.

2.1.20 Incorrect Handling of Range List Retrieval in Optimizer

The following query should apply [(2:2)3] instead of [1:1].

```
set flags 'strategy,detail';

for r in oropt_rel
   with r.fld1 = 'a1' and r.fld2 = 'b1' and r.fld3 = 'c1'
     or r.fld1 = 'a1' and r.fld2 = 'b2' and r.fld3 = 'c2'
     or r.fld1 = 'a1' and r.fld2 = 'b3' and r.fld3 = 'c3'
   print r.fld1, r.fld2, r.fld3
end_for
Leaf#01 FFirst OROPT_REL Card=1025
  BgrNdx1 OROPT_REL_INDEX1 [(1:1)3] Fan=18
  BgrNdx2 OROPT_REL_INDEX2 [1:1] Fan=15 <== should be [(2:2)3] Bool
```

This problem was detected only in internal testing due to a change in the order of indices OROPT_REL_INDEX1 and OROPT_REL_INDEX2 in terms of the physical address created in the memory.

The first CRTV of the common segments is incorrectly configured and thus [1:1] is selected over [(2:2)3] .

There is no known workaround for this problem.

This problem has been corrected in Oracle Rdb Release 7.0.6.4.

2.1.21 Bugchecks at DIOCCH$FETCH_SNAP_SEG + 00000594

Bug 1879372

In rare cases of relatively high system load with intensive access to cached records between read−write and read−only processes, it was possible for a read−only process to fail with an exception at DIOCCH$FETCH_SNAP_SEG + 00000594.

This bugcheck was due to incorrect memory access ordering. Read−only processes would sometimes get an incorrect snapshot page pointer and find that the snapshot page was not for the matching live page.

The chance of seeing this bugcheck has been reduced in Oracle Rdb Release 7.0.6.4.

2.1.22 IVP Tests of DECC and VAXC

Previously, depending on the versions of and existence/absence of the :VAX C and DEC C compilers, the IVP may not have correctly tested one or both of the compiler options.

This particular problem area has been improved. The Oracle Rdb IVP procedure now attempts to more accurately determine what C compilers have been installed. However, Oracle is not able to test every combination of every version of the VAXC and DECC compilers; it is possible that the SQL precompiler IVP may have difficulty with certain combinations of versions. Oracle believes the IVP will work correctly in

most cases.

2.1.23 Left OJ Query Using Hash Retrieval Returns Wrong Results

Bug 2352298

The following left OJ query should not select NULLs for the columns of the right outer join table.

```
create database filename 'testdb.RDB'
    create storage area AREA_T2 filename 'AREA_T2.RDA'
        page format is MIXED;

create table T1 (T1_COL_A CHAR(3), T1_COL_C CHAR(6) );
create table T2 (T2_COL_A CHAR(3), T2_COL_B CHAR(2), T2_COL_C CHAR(6) );

create unique index T1_AC_SRT on T1 (T1_COL_A, T1_COL_C) type is SORTED;

create unique index T2_ABC_HSH on T2 (T2_COL_A, T2_COL_B, T2_COL_C)
    type is HASHED store in AREA_T2;

create unique index T2_ACB_SRT on T2 (T2_COL_A, T2_COL_C, T2_COL_B)
    type is SORTED;

create storage map MAP_T2 for T2
    placement via index T2_ABC_HSH store in AREA_T2;

insert into T1 (T1_COL_A, T1_COL_C) value ('ABC', '123456');

insert into T2 (T2_COL_A, T2_COL_B, T2_COL_C) value
('ABC', '01', '123456');

set flags 'strategy,detail';

select T1_COL_A, T1_COL_C,
       T2_COL_A, T2_COL_B, T2_COL_C from
   (T1 left outer join T2 on T2_COL_A   = T1_COL_A and
                             T2_COL_B   = '01'     and
                             T2_COL_C   = T1_COL_C)
  where T1_COL_A = 'ABC' and
        T1_COL_C = '123456';
Tables:
  0 = T1
  1 = T2
Merge of 1 entries
  Merge block entry 1
  Conjunct: (0.T1_COL_A = 'ABC') AND (0.T1_COL_C = '123456')
  Cross block of 2 entries        (Left Outer Join)
    Cross block entry 1
      Conjunct: (0.T1_COL_A = 'ABC') AND (0.T1_COL_C = '123456')
      Get     Retrieval by index of relation 0:T1
        Index name  T1_SRT [2:2]       Direct lookup
          Keys: (0.T1_COL_A = 'ABC') AND (0.T1_COL_C = '123456')
    Cross block entry 2
      Conjunct: (1.T2_COL_A = 0.T1_COL_A) AND (1.T2_COL_B = '01') AND
                (1.T2_COL_C = 0.T1_COL_C)
      Get     Retrieval by index of relation 1:T2
        Index name  T2_ABC_HSH [2:2]            <== Note1
          Keys: (1.T2_COL_A = 0.T1_COL_A) AND (1.T2_COL_B = '01')
 T1_COL_A   T1_COL_C     T2_COL_A   T2_COL_B   T2_COL_C
 ABC        123456       NULL       NULL       NULL
1 row selected
```

Note 1: Partial Index retrieval [2:2] using 2 segments is applied to the hash index of 3 segments and no row is found.

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

1. The main query contains a derived table of a left outer join on the columns COL_A and COL_C between tables T1 and T2 but T2_COL_B is equal to a literal '01'.
2. The WHERE clause contains the filter predicates on COL_A and COL_C of table T1.
3. The outer table T1 has a sorted index, T1_AC_SRT, with columns COL_A and COL_C.
4. The inner table T2 has a hash index, T2_ABC_HSH, with columns COL_A, COL_B, and COL_C, and another sorted index, T2_ACB_SRT, with columns COL_A, COL_C, and COL_B.
5. Having this sorted index, T2_ACB_SRT, exposes the hidden bug in the optimizer where the partial index retrieval [2:2] using 2 segments out of 3 on a hash index, returns wrong results.

As a workaround, the query works if the sort index T2_ACB_SRT is dropped.

Here is the new strategy and result:

```
Tables:
  0 = T1
  1 = T2
Merge of 1 entries
  Merge block entry 1
  Conjunct: (0.T1_COL_A = 'ABC') AND (0.T1_COL_C = '123456')
  Cross block of 2 entries        (Left Outer Join)
    Cross block entry 1
      Conjunct: (0.T1_COL_A = 'ABC') AND (0.T1_COL_C = '123456')
      Get     Retrieval by index of relation 0:T1
        Index name  T1_SRT [2:2]       Direct lookup
          Keys: (0.T1_COL_A = 'ABC') AND (0.T1_COL_C = '123456')
    Cross block entry 2
      Conjunct: (1.T2_COL_A = 0.T1_COL_A) AND (1.T2_COL_B = '01') AND
                (1.T2_COL_C = 0.T1_COL_C)
      Get     Retrieval by index of relation 1:T2
        Index name  T2_ABC_HSH [3:3]       Direct lookup <== Note2
          Keys: (1.T2_COL_A = 0.T1_COL_A) AND (1.T2_COL_B = '01') AND
                (1.T2_COL_C = 0.T1_COL_C)
 T1_COL_A   T1_COL_C     T2_COL_A   T2_COL_B   T2_COL_C
 ABC        123456       ABC        01         123456
1 row selected
```

Note 2: Index retrieval [3:3] using all segments is applied to the hash index of 3 segments and the correct row is found.

This problem has been corrected in Oracle Rdb Release 7.0.6.4.

2.1.24 Getting Null Values Instead of Actual Values

Bug 2245379

When columns of a table have been added/dropped several times, it is possible, in some rare conditions, to get null values instead of actual values for a column when doing a sequential scan of the table.

The following example shows the different results depending on the strategy.

```
SQL> select f1,f2 from t where f1>0 and f1<3;
Index only retrieval of relation T
  Index name  I_T [1:1]
```

```
          F1            F2
           1             1
           2             1

SQL> select f1,f2,f3 from t where f1>0 and f1<3;
Conjunct        Get     Retrieval sequentially of relation T
          F1            F2            F3
           1          NULL             1
           2          NULL             1
```

To work around the problem, add a new column to the table.

```
SQL> Alter table t add column xx integer;
SQL> commit;
```

This problem has been corrected in Oracle Rdb Release 7.0.6.4.

2.2 SQL Errors Fixed

2.2.1 ALTER DOMAIN...DROP DEFAULT Reports DEFVALUNS Error

Bug 456867

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

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

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

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

This problem has been corrected in Oracle Rdb Release 7.0.6.4.

2.2.2 Exception in Nested Routines Did Not Close Index Scans

Bug 1058528

In prior releases of Oracle Rdb, exceptions occurring in a nested stored procedure or function could leave open index scans. This problem resulted in bugchecks during the ROLLBACK statement. The bugcheck would have a signature similar to the following:

```
***** Exception at 03435E80 : KOD$ROLLBACK + 000001D8
%COSI-F-BUGCHECK, internal consistency failure
```

To cause this problem, an exception had to be raised while a query was scanning an index. A nested stored routine is one invoked by the CALL statement or a function expression within a compound statement (BEGIN END block).

The exception may occur in the routine body itself, a trigger action, or in a nested stored routine. There is no workaround for this problem.

This problem was corrected in Oracle Rdb Release 7.0.6 but the note was lost somewhere along the line.

Note

The routine KOD$ROLLBACK implements the rollback functionality. It contains a sanity check to ensure correct operation of the Rdb server. While every effort has been made to correct this reported problem, it is possible that some unrelated problem may produce a

2.2.3 DDL Statements Generated Unexpected Runtime Errors

In previous releases of Oracle Rdb, it was possible for DDL (data definition language) statements embedded in the SQL precompiler source (EXEC SQL) or in a SQL module language procedure to generate unexpected errors at run time. This problem only occurred when the quote character (') had to be doubled when included in a string literal.

Consider this CREATE TABLE example embedded in a C source module:

```
void sql_signal ();
main()
{
int SQLCODE = 0;
exec sql declare alias filename 'MF_PERSONNEL';
exec sql create table my_table1 (name_q char(10) default '''');
if (SQLCODE != 0) sql_signal ();
exec sql rollback;
if (SQLCODE != 0) sql_signal ();
}
```

When this application is executed, the following error is reported:

```
%SQL-F-UNTSTR, Unterminated string found
```

The problem occurs because all DDL statements (such as CREATE TABLE) are processed as Dynamic statements by SQL module language and the SQL precompiler. The saved version of the CREATE TABLE statement is rewritten without processing the quoting character (') correctly.

In most cases, this problem would cause SQL-F-SYNTAX_ERR or %SQL-F-UNTSTR exceptions, but in some cases two mismatched quotes may have unexpectedly captured syntax and the statement may have executed correctly. See the following TRACE statement in a stored procedure for instance:

```
TRACE '''' || R.RDB$FIELD_NAME || '''';
```

This statement was saved as:

```
TRACE ''' || R . RDB$FIELD_NAME || ''';
```

which caused Trace to display the following text:

```
~Xt: ' || R . RDB$FIELD_NAME || '
```

This problem has been corrected in Oracle Rdb Release 7.0.6.4. SQL now encodes the quoted string correctly for use with Dynamic SQL. Any applications that suffer from this problem must be recompiled using the corrected version of Rdb.

2.2.4 INSERT Cursor on a Derived Table Would Bugcheck

In prior releases of Oracle Rdb, SQL did not prevent a derived table from being used as the target for an INSERT cursor. While the DECLARE and OPEN for the cursor apparently succeeded, attempts to use the cursor would generate a bugcheck as shown below.

```
SQL> declare ONE insert only table cursor
```

```
cont> for select EMPLOYEE_ID from (select * from EMPLOYEES) as E;
SQL> OPEN ONE;
SQL> INSERT INTO CURSOR ONE VALUES ('00000');
%SQL-I-BUGCHKDMP, generating bugcheck dump file DISK1:[TESTING]SQLBUGCHK.DMP;
%SYSTEM-F-ACCVIO, access violation, reason mask=00, virtual address=00000024,
PC=00239A72, PSL=03C00005
```

This problem has been corrected in Oracle Rdb Release 7.0.6.4. SQL now issues an error when the DECLARE CURSOR is detected.

2.2.5 SQL Query Bugchecks at SQL$$GET_QUEUE_WALK

Bug 2245763

In prior releases of Oracle Rdb, it was possible that some queries involving UNION and functions that returned VARCHAR would bugcheck when using ORACLE LEVEL1 dialect.

```
***** Exception at 002A5844 : SQL$$GET_QUEUE_WALK + 00000244
%SYSTEM-F-ACCVIO, access violation, reason mask=00,
virtual address=0000000000000080, PC=00000000002A5844, PS=0000001B
```

The following example shows this problem.

```
SQL> set dialect 'oracle level1';
SQL>
SQL> create module MMM
cont>     language SQL
cont>     function rtrim (in :a varchar (200), in: c varchar (200))
cont>     returns varchar(200);
cont>     return trim (both :c from :a);
cont> end module;
SQL>
SQL> select ' '
cont> from employees e
cont> union
cont> select rtrim(e.first_name,' ')
cont> from employees e;
%SQL-I-BUGCHKDMP, generating bugcheck dump file DISK1:[TEST_DB]SQLBUGCHK.DMP;
%SYSTEM-F-ACCVIO, access violation, reason mask=00,
virtual address=0000000000000080, PC=00000000002A5844, PS=0000001B
SQL>
```

This problem was caused by erroneous processing of the implicit CASE expression wrapped around the function call to produce Oracle RDBMS language semantics for zero length strings which are considered equivalent to NULL.

A workaround would be to use SET DIALECT 'SQL92' before executing this query. In this dialect, no special zero length string handling is required.

This problem has been corrected in Oracle Rdb Release 7.0.6.4.

2.2.6 SQL Query Bugchecks at SQL$$GET_QUEUE_WALK

Bug 2272808

In prior releases of Oracle Rdb, it was possible that some queries involving UNION, COALESCE (or NVL) builtin functions would bugcheck.

```
***** Exception at 003363D0 : SQL$$GET_QUEUE_WALK + 00000340
%SYSTEM-F-ACCVIO, access violation, reason mask=00,
virtual address=0000000000000080, PC=00000000003363D0, PS=0000001B
```

The following example shows this problem.

```
SQL> create module MMM
cont>     language SQL
cont>     function fixstr (in :id integer,
cont>                 in :a char (20),
cont>                 in :b char (20),
cont>                 in :c char (20))
cont>     returns char(20);
cont>     return NULL;
cont> end module;
SQL>
SQL> select fixstr (1, last_name, first_name, middle_initial) as nm
cont>  from employees
cont>  where employee_id = '00164'
cont> union all
cont> select cast(coalesce(postal_code,
cont>                fixstr (1, last_name, first_name, middle_initial)
cont>               ) as char(20)) as nm
cont>  from employees
cont>  where employee_id = '00164';
%RDMS-I-BUGCHKDMP, generating bugcheck dump file DISK1:[TEST_DB]SQLBUGCHK.DMP;
%SYSTEM-F-ACCVIO, access violation, reason mask=00,
virtual address=0000000000000080, PC=00000000003363D0, PS=0000001B
```

This problem was caused by erroneous processing of the COALESCE expression wrapped around the function call in the second leg of the UNION clause.

Note

The NVL function is a synonym for COALESCE.

A workaround would be to rewrite the COALESCE as a searched case expression. *COALESCE (a, b, ..., z)* is equivalent to:

```
case
    when a is not NULL then a
    when b is not NULL then b
    ...
    else z
end
```

This problem has been corrected in Oracle Rdb Release 7.0.6.4.

2.2.7 Privileges Not Honored For SET TRANSACTION

Bug 1668270

Oracle Rdb was not reverting to the privilege settings of the SQL/Services service owner for commands such as SET TRANSACTION ... RESERVING, CREATE, ALTER and DROP.

The following example uses SQL*Plus and SQL*Net for Rdb to execute a query and shows this behaviour.

```
SQL> set transaction read write reserving employees for protected write;
set transaction read write reserving employees for protected write;
```

```
*
ERROR at line 1:
ORA-01031: insufficient privileges
```

In fact, the current user is granted access to the EMPLOYEES table, but the service owner is not. A workaround is to give the service owner the required privileges.

This problem has been corrected in Oracle Rdb Release 7.0.6.4.

2.2.8 Multistatement Procedures Used with Connections Resulted in %RDB-E-OBSOLETE_METADA Error Message

Bug 1879521

In prior releases of Oracle Rdb, there was a problem with multiple connections and the use of multistatement procedures. Specifically, Oracle Rdb requires a special internal module to be set up for multistatement procedures. In the case of two or more connections calling the same multistatement procedure, the module setup was not done for the second connection. This was incorrect behavior and resulted in the following error message:

```
%RDB-E-OBSOLETE_METADA, request references metadata objects that no longer
exist
```

The correct behavior is to insure that the module setup is performed when a database switch occurs for the first time.

This problem has been corrected in Oracle Rdb Release 7.0.6.4.

2.2.9 Incorrect Handling of FOR Loop Select List Columns

Bug 2309767

In prior releases of Oracle Rdb, reference to the DBKEY of a table using the FOR loop variable could return the wrong table's DBKEY.

The following example shows this problem.

```
SQL> begin
cont> declare :rc integer = 0;
cont>
cont> for :EJH as
cont>    select JH.dbkey
cont>    from employees E, job_history JH
cont>    where jh.employee_id = '00164'
cont>    and jh.employee_id = e.employee_id
cont> do
cont>   trace 'processing row';
cont>   update job_history JH
cont>    set jh.employee_id = '00164'
cont>    where JH.DBKEY = :EJH.DBKEY;
cont>   get diagnostics :rc = row_count;
cont>   trace 'updated ', :rc;
cont> end for;
cont> end;
%RDB-E-NO_RECORD, access by dbkey failed because dbkey is no longer associated
with a record
-RDMS-F-NODBK, 66:15:1 does not point to a data record
```

A FOR loop declares a special variable which can be used to reference the results of the select statement. This list is used to match the names for subsequent references in the FOR loop body. The DBKEY of a table is an exception as it need not be explicitly selected from the table. It was this exception which caused the incorrect behavior.

The following problems are corrected in this release of Rdb.

1. FOR :A AS SELECT DBKEY FROM T
   In prior releases, the select list was not used to resolve the DBKEY reference. Instead, the first DBKEY found for all tables in the join was used. SQL now uses the select list first to locate the DBKEY. If none is found and the FOR loop is processing a single table, then that table's DBKEY is used. If more than one DBKEY is visible because of a join condition, then an error will be reported.

   ```
   %SQL-F-FLDAMBIG, Column DBKEY is not unique in tables in the FROM clause
   ```
2. FOR :A AS SELECT LAST_NAME AS DBKEY FROM T
   While not recommend by Oracle, it is legal to rename a column as "DBKEY" using the AS clause. In prior releases, this renaming was ignored and the actual DBKEY of the table was used. SQL now processes the renamed column correctly.
3. FOR :A AS SELECT ROWID FROM T
   The names DBKEY and ROWID are synonymous. It is possible to fetch the ROWID in the select list and later reference the value using DBKEY. Oracle recommends that the names be used consistently as it is possible that this behavior may change in a future release.
4. FOR :A AS SELECT * FROM T, S
   In prior releases, a reference to DBKEY in a join context was allowed and SQL would choose the first table. This erroneous behavior is now fixed. SQL will now report an error.

   ```
   %SQL-F-FLDAMBIG, Column DBKEY is not unique in tables in the FROM clause
   ```
5. FOR :A AS SELECT * FROM EMPLOYEES, JOB_HISTORY
   If columns from different tables have the same name or if the AS clause is used to rename a column to the same name as an existing column, then SQL did not report ambiguous column references within the FOR loop body.

   ```
   SQL> begin
   cont> declare :rc integer = 0;
   cont>
   cont> for :EJH as
   cont>    select *
   cont>    from employees E, job_history JH
   cont>    where jh.employee_id = '00164'
   cont>    and jh.employee_id = e.employee_id
   cont> do
   cont>   trace 'processing row';
   cont>   update job_history JH
   cont>    set jh.employee_id = '00164'
   cont>    where JH.employee_id = :EJH.employee_id;
   cont>   get diagnostics :rc = row_count;
   cont>   trace 'updated ', :rc;
   cont> end for;
   cont> end;
   %SQL-F-FLDAMBIG, Column EMPLOYEE_ID is not unique in tables in the FROM clause
   ```

These problems have been corrected in Oracle Rdb Release 7.0.6.4. Please note that existing applications (stored procedures and compiled programs) will continue to function. However, when they next are compiled (or created), this new error checking will be in effect.

2.2.10 Unexpected Error on FOR Loop With Dialect ORACLE LEVEL1

In prior releases of Oracle Rdb, Dynamic SQL would generate an error if a FOR loop was detected and the dialect was set to ORACLE LEVEL1. The errors could occur for a stored procedure (part of a CREATE MODULE statement), or a compound statement.

The following examples show these errors. The first error shown is for a dynamically executed compound statement.

```
>> ATTACH 'filename DB$:SCRATCH'
>> SET DIALECT 'ORACLE LEVEL1'
>> SET FLAGS 'TRACE'
>> BEGIN FOR :C AS SELECT RDB$FLAGS FROM RDB$DATABASE DO TRACE :C.RDB$FLAGS;
END FOR; END
error: -1...
error text:
%SQL-F-DATTYPUNK, Data type unknown. Expression cannot use only host variables
```

The second error was generated from a compiled SQL$PRE application that included CREATE MODULE as part of an EXEC SQL statement. The source of the CREATE MODULE is passed to Dynamic SQL for execution.

```
exec sql
    create module MYMOD
     language SQL
    procedure MYPROC (in :a integer);
    begin
    declare :b float = 0;
    for :c as select rdb$flags from rdb$database
    do
        trace :c.rdb$flags, :b, :a;
    end for;
    end;
    end module;
```

This runtime error is produced:

```
%SQL-F-DDLPARAM, You referred to parameter ? in a DDL statement
```

These errors occur because the FOR loop variable is erroneously assumed to be a parameter marker for Dynamic SQL.

This problem has been corrected in Oracle Rdb Release 7.0.6.4. SQL now recognizes FOR loop variables as well as routine parameters and declared variables and no longer assumes they are parameter markers.

2.2.11 Unexpected Truncation of Data Assigned in Precompiled SQL

Bug 968518

In prior releases of Oracle Rdb, it was possible that multiple UPDATE or INSERT statements in the same multistatement procedure using the same host variable would assign truncated string values to some columns.

A SQL optimization tried to limit the passed data to that required by the column. For example, if the host variable was longer than the target column then data passed to the Rdb server was limited to the column

length (since it would be truncated during assignment anyway). If the same host variable was assigned to columns of differing lengths then it was possible that the smaller length would be used for both assignments.

A workaround is to reorder the INSERT or UPDATE statements.

This problem has been corrected in Oracle Rdb Release 7.0.6.4. The SQL precompiler now ensures that the larger target length is used for the assignment.

2.3 Oracle RMU Errors Fixed

2.3.1 RMU /VERIFY /ROOT Incorrectly Reports RMU−E−BADAIJPN and/or RMU−E−AIJNOTFND

Previously, it was possible for the RMU /VERIFY /ROOT command to incorrectly attempt to access a non−existant after−image journal file. This problem was caused by an incorrect bounds check that resulted in one additional, non−existant, internal data structure being used. In very rare cases, this data structure appeared to contain an incorrect (or blank) name of an after−image journal file.

For example, the follow error might be displayed (note the two spaces between "file" and "not" in the second message; this is where the filename would typically be displayed − in this case the name was blank):

```
$ RMU /VERIFY /ROOT THUNDER.RDB
%RMU−E−BADAIJPN, There is no name associated with AIJ entry 1.
%RMU−E−AIJNOTFND, expected after−image file  not found
%RMU−W−ROOERRORS, 1 error encountered in root verification
```

This problem has been corrected in Oracle Rdb Release 7.0.6.4. The RMU /VERIFY utility now only checks the valid internal data structure for after−image journal files.

2.3.2 RMU /VERIFY Index Warnings Eliminated on Small Cardinality

A problem with the way cardinality differences between index nodes within sorted ranked indexes were determined caused RMU /VERIFY to log spurious cardinality differences warnings when verifying sorted ranked indexes.

Small differences in cardinalities, when the value of the cardinalities were small, would be incorrectly logged as warnings during the index verify.

An example of such a warning is show below.

```
%RMU−W−BTRLEACAR, Inconsistent leaf cardinality (C2) of 3
    specified for entry 12 at dbkey 78:92351:1 using precision of 33.
Dbkey 78:216139:1 at level 2 specified a cardinality of 1
```

These warnings should be ignored if both cardinalities involved are very small, say less than 10.

This problem has been corrected in Oracle Rdb Release 7.0.6.4. RMU /VERIFY will no longer log a warning when both the cardinality differences and their values are small.

2.3.3 RMU /VERIFY /CONSTRAINT Now Uses Warning for CONSTFAIL Message

Enhancement Bug 1644732

In previous releases of Oracle Rdb, RMU /VERIFY /CONSTRAINT would issue an informational message if a constraint failed to verify correctly. This severity was often ignored by log file summarizers and so the severity of the CONSTFAIL message has been changed to a warning as shown in the following example.

```
$ RMU /VERIFY /CONSTRAINT SQL$DATABASE
%RMU-W-CONSTFAIL, Verification of constraint "T_CHECK1" has failed.
%RMU-W-CONSTFAIL, Verification of constraint "T_CHECK2" has failed.
```

This problem has been corrected in Oracle Rdb Release 7.0.6.4.

2.3.4 RMU Incremental Backup and Restore Could Cause Truncated Table Rows to Reappear

Bugs 1926428 and 1987848

There was a problem with RMU /BACKUP /INCREMENTAL and RMU /RESTORE /INCREMENTAL where rows deleted by a truncate table command in SQL could reappear following an incremental RMU /RESTORE of uniform storage areas where a truncate table operation had taken place since the last full backup. This happened because RMU /BACKUP /INCREMENTAL and RMU /RESTORE /INCREMENTAL did not save and restore the status of deleted rows from truncated tables as having been deleted due to a truncate table operation.

This problem has been corrected in Oracle Rdb Release 7.0.6.4.

2.3.5 Deleted Rows Reappear After RMU /REPAIR

Bug 1926428

If a table was truncated and then the RMU /REPAIR /INIT=FREE or RMU /REPAIR /SPAM command was executed, the rows deleted by the truncate command could reappear making the database unreliable.

This problem has been corrected in Oracle Rdb Release 7.0.6.4.

Note that /SPAM is the default qualifier on the RMU /REPAIR command only if none of the following qualifiers is specified on the RMU /REPAIR command line:

- /AIP
- /ABM
- /INITIALIZE = FREE_PAGES
- /INITIALIZE = SNAPSHOTS
- /INITIALIZE = SNAPSHOTS = CONFIRM

Previously /SPAM was the default qualifier on all RMU /REPAIR commands.

2.3.6 RMU /COLLECT OPTIMIZER_STATISTICS Fails When Temporary Tables in Database

Bug 2245491

In previous releases of Oracle Rdb, the RMU /COLLECT OPTIMIZER_STATISTICS command would fail if there were temporary tables in the database that also had storage maps defined. The storage maps can be used to disable compression as shown in this example.

```
SQL> create global temporary table GT (a integer);
SQL> create storage map GT_MAP for GT
cont>    disable compression;
```

When this database was processed using RMU /COLLECT the following error would occur:

```
$ RMU /COLLECT /LOG OPTIMIZER_STATISTICS TEST_DB
Start loading tables... at 21-MAR-2002 14:39:50.24
%SYSTEM-F-ACCVIO, access violation, reason mask=04, virtual
address=0000000000000068, PC=0000000000345B14, PS=0000001B
%RMU-F-FATALOSI, Fatal error from the Operating System Interface.
%RMU-I-BUGCHKDMP, generating bugcheck dump file DISK1:[TEST_DB]RMUBUGCHK.DMP;
%RMU-F-FTL_ANA, Fatal error for ANALYZE operation at 21-MAR-2002 14:39:50.85
```

A workaround for this problem is to drop just the storage maps for the temporary tables. RMU /COLLECT normally ignores views and temporary tables. Once the RMU /COLLECT command has been executed, the storage maps can be re-created for the temporary tables.

This problem has been corrected in Oracle Rdb Release 7.0.6.4. RMU /COLLECT now correctly filters temporary tables that also have storage maps.

2.3.7 RMU /BACKUP /LOADER_SYNCH and Usage of Tape Labels

Bug 1914113

The RMU /BACKUP commands offer concurrent tape drive operation. The qualifier /LOADER_SYNCHRONIZATION allows users to preload tapes to minimize operator support. The backup operation goes fine without any operator support if all the threads/drives use the same number of volumes. One disadvantage with using the /LOADER_SYNCHRONIZATION qualifier is that, because not all threads backup equal volumes of data, some operator support may or may not be required to load the tapes in stages as backup threads become inactive (refer to "Guide to Database Maintenance" for more information). Using implicit tape labeling or the qualifier /ACCEPT_LABEL may not be acceptable to some users for label security reasons. In the above scenario, an operator is required to monitor the backup operation throughout its course so that the backup is completed in time.

To handle the above situation without operator intervention, the qualifier /LOADER_SYNCHRONIZATION will now accept a keyword "FIXED" which, when specified, forces the assignment of the labels to the drives regardless of how many tapes a thread actually uses.

For example, assume that a user uses three tape drives, each having its own loader, for RMU /BACKUP. Each tape drive is preloaded with two tapes in the following distribution:

```
Drive           I               II              II
Label       tape01          tape02          tape03
            Tape04          tape05          tape06
```

The user can use the following RMU /BACKUP command to force the assignments of the labels to the drives irrespective of the number of tapes used by each drive.

```
$ RMU /BACKUP /LOADER_SYNCHRONIZATION=FIXED -
  /LABEL=(TAPE01,TAPE02,TAPE03,TAPE04) -
  {DATABASE_TO_BE_BACKED_UP} -
  DRIVE1:{BACKUP_FILE_NAME}, DRIVE2:, DRIVE3:
```

This problem has been corrected in Oracle Rdb Release 7.0.6.4.

2.3.8 RMU /BACKUP to Tape can Hang on a Quit Response to a Prompt

Bug 2303545

On an RMU /BACKUP to tape, when the user specified the "QUIT" response to a prompt indicating that the backup should terminate and not complete because the wrong tape was mounted or for some other reason, the backup threads could hang while RMU /BACKUP was terminating. This was actually part of a larger problem which caused backup thread hangs when a fatal error was signaled. The QUIT prompt signals a fatal error, "%RMU−F−ABORT, operator requested abort on fatal error" to terminate the backup.

The following example shows that RMU /BACKUP to tape could hang when a QUIT response was given to an RMU prompt to a user terminal or the operator console indicating that the backup operation should terminate and not continue.

```
$  RMU /BACKUP /LOG /ONLINE /LABEL=TEST1 /NOREWIND /DENSITY=1 −
     MF_PERSONNEL TAPE_DEVICE:MF_PERSONNEL

%RMU−E−FATALERR, fatal error on tape_device:[000000]MF_PERSONNEL.RBF;
−SYSTEM−F−VOLINV, volume is not software enabled

%RMU−I−SPECIFYC, specify option (QUIT or CONTINUE)

RMU> QUIT

%RMU−F−ABORT, operator requested abort on fatal error
```

The only way to avoid this problem is to correct the cause of the RMU prompt so the prompt will not be output.

This problem has been corrected in Oracle Rdb Release 7.0.6.4.

2.3.9 RMU /BACKUP to Tape can Hang When Terminating on Fatal Errors

Bug 2303952

On an RMU /BACKUP to tape, when a fatal error was signaled, a hang could occur. The hang occurred when RMU /BACKUP was attempting to terminate because of the fatal error. This problem was due to a problem handling fatal errors.

The following example shows that RMU /BACKUP to tape could hang when it was attempting to terminate due to a fatal error.

```
$ RMU /BACKUP /LOG /CHECKSUM /LABEL=BACK01 /NOREWIND −
    MF_PERSONNEL TAPE_DEVICE:PERSONNEL.RBF

%RMU−I−BCKTXT_02, Full backup of storage area (RESUMES)
DEVICE:[DIRECTORY]RESUMES.RDA;1

%RMU−F−CANTREADDBS, error reading pages 2:15−15
−RMU−F−CHECKSUM, checksum error − computed F79F2744, page contained F7A72744
```

The only way to avoid this problem is to correct the cause of the fatal error.

This problem has been corrected in Oracle Rdb Release 7.0.6.4.

2.3.10 /LOG Qualifier Default for RMU /SET LOGMINER and RMU /SET ROW_CACHE

Previously, the default setting for the "/LOG" qualifier for the RMU /SET LOGMINER and RMU /SET ROW_CACHE commands was incorrect. The default setting for the "/LOG" qualifier should be the process' current default DCL VERIFY state. The "/LOG" qualifier incorrectly defaulted to display log messages.

This problem has been corrected in Oracle Rdb Release 7.0.6.4. The log message default state now tracks a process' current default DCL VERIFY state.

2.3.11 RMU /UNLOAD /AFTER_JOURNAL Exception in AIJEXT$FINISH

Previously, it was possible for the RMU /UNLOAD /AFTER_JOURNAL command to loop while writing bugcheck dump files as the result of exception. Typically, the dump files would all have an exception in the AIJEXT$FINISH routine.

This problem has been corrected in Oracle Rdb Release 7.0.6.4. The AIJEXT$FINISH routine now checks to make sure that certain data structures are initialized before using them.

2.3.12 RMU /UNLOAD /AFTER_JOURNAL Wildcard Table Names

The RMU /UNLOAD /AFTER_JOURNAL command now supports wildcard processing of table names. The asterisk (*) and the percent sign (%) wildcard characters can be used in the table name specification to select all tables that satisfy the components you specify. The asterisk matches zero or more characters and the percent sign matches a single character.

For table name specifications that contain wildcard characters, if the first character of the string is a pound sign (#), the wildcard specification is changed to a not matching comparison. This allows exclusion of tables based on a wildcard specification. The pound sign designation is only evaluated when the table name specification contains an asterisk or percent sign.

For example, a table name specification of * indicates that all tables in the database are to be selected. A table name specification of "FOO%" indicates that all table names that are four characters long and that start with the string "FOO" (such as "FOOD" and "FOOT") are to be selected.

A table name specification of "#*FOO*" specifies that all table names that *do not* contain the string "FOO" (excluding those tables such as "FOOD", "SEAFOOD" and "BUFFOONS") are to be selected.

This change has been made in Oracle Rdb Release 7.0.6.4.

2.3.13 RMU /UNLOAD /AFTER_JOURNAL AIJ Backup and Restart Information

Previously, the next backup file from after a quiet−point AIJ backup was always required to be the first one supplied to the LogMiner. However, when restart information is present, an internal quiet−point can be implied so long as the first AIJ backup specified is prior to the backup sequence number indicated in the restart information. Because of this, when restart information is supplied, the actual check for the quiet point backup can be waived.

This change has been made in Oracle Rdb Release 7.0.6.4.

2.3.14 Unexpected COSI−F−TRU Error from RMU Extract

Bug 2349013

In prior releases, RMU Extract could fail with an error while processing complex view definitions.

```
$ RMU /EXTRACT /ITEM=VIEW /OUTPUT=VIEWS.SQL TESTDB
%COSI−F−TRU, truncation
%RMU−F−FATALOSI, Fatal error from the Operating System Interface.
%RMU−F−FTL_RMU, Fatal error for RMU operation at 29−APR−2002 12:44:10.71
```

This problem has been corrected in Oracle Rdb Release 7.0.6.4.

2.4 RMU Show Statistics Errors Fixed

2.4.1 RMU /SHOW STATISTICS Triggers Invoked From User Defined Events at Times Other Than the Refresh Intervals

Bug 2158913

RMU /SHOW STATISTICS triggers can invoke from a user defined event at times other than refresh intervals. Moreover, the invoke is triggered more than once for each time the threshold is reached. The same event works fine when a "NOTIFY" is used instead of an "INVOKE".

This problem has been corrected in Oracle Rdb Release 7.0.6.4. At present (up to releases 7.0.6.3 and 7.1.0.1), the RMU /SHOW STATISTICS display is updated when the RMU /SHOW STATISTICS keypad is used apart from being updated at refresh intervals. As a result of the fix for this problem, RMU /SHOW STATISTICS display will only be updated at refresh intervals.

2.4.2 RMU /SHOW STATISTICS Row Cache Information May Not Display the Information of the Cache Selected

Bugs 2220998 and 2150808

RMU /SHOW STATISTICS may not display the information about the correct cache when you select the "Row Cache Information" option.

This problem has been corrected in Oracle Rdb Release 7.0.6.4.

2.4.3 Inconsistency in the Hot Standby Statistics Screen of RMU /SHOW STATISTICS

Bug 1943101

An inconsistency is observed on the Hot Standby Statistics screen of RMU /SHOW STATISTICS. On the standby side, the master AIJ seems smaller than the standby AIJ.

This problem has been corrected in Oracle Rdb Release 7.0.6.4.

Chapter 3
Software Errors Fixed in Oracle Rdb Release 7.0.6.3

This chapter describes software errors that are fixed by Oracle Rdb Release 7.0.6.3.

3.1 Software Errors Fixed That Apply to All Interfaces

3.1.1 Disabling AIJ When Row Cache Recovery Required

Bug 1831040

When after−image journaling is manually disabled on a closed database that had Row Caching active and requires recovery, it is possible to render the database unusable. For example, consider the following sequence of events:

1. Database is running with Row Caching enabled.
2. AIJ files not backed up and eventually fill.
3. User processes deleted or system fails.
4. User enters RMU /SET AFTER_JOURNAL /DISABLE command.

At this point, a warning message is displayed, but the database can not be opened because the DBR process will fail when attempting to access the after image journal files.

This problem has been corrected in Oracle Rdb Release 7.0.6.3. Attempts to disable journaling will now result in a fatal error and journaling will not be disabled when Row Cache recovery is required. The following example demonstrates this condition.

```
$ RMU/SET AFTER/DISABLE MF_PERSONNEL.RDB
%RMU−W−DBRABORTED, database recovery process terminated abnormally
%RMU−F−MUSTRECDB, database must be closed or recovered
%RMU−F−FTL_SET, Fatal error for SET operation at 11−SEP−2001 22:52:22.37
```

3.1.2 Query With Range List OR Predicates Returns Wrong Results

Bug 1329838

The following query with range list OR predicates returns wrong results:

```
set flags 'strategy,detail';

select t,m,p,b from a
  where (t='S' and (m='N' or p='Q')) or (t='Z' and (m='N' or b='A'))
  order by t,m,p,b;
Tables:
  0 = A
Sort: 0.T(a), 0.M(a), 0.P(a), 0.B(a)
Conjunct: ((0.T = 'S') AND ((0.M = 'N') OR (0.P = 'Q'))) OR ((0.T = 'Z') AND ((
          0.M = 'N') OR (0.B = 'A')))
OR index retrieval                          ! <== Let's call this "Outer"
  Conjunct: (0.B = 'A') OR (0.M = 'N') OR (0.M = 'N')
  OR index retrieval                        ! <== let's call this "Inner"
    Get     Retrieval by index of relation 0:A
      Index name  BTY_X [2:2]
        Keys: (0.B = 'A') AND (0.T = 'Z')
    Conjunct: NOT (0.B = 'A') AND ((0.M = 'N') OR (0.M = 'N')) ! <== incorrect
    Get     Retrieval by index of relation 0:A
      Index name  MTZ_X [(2:2)2]
        Keys: r0: (0.M = 'N') AND (0.T = 'S')
```

```
             r1: (0.M = 'N') AND (0.T = 'Z')
  Conjunct: NOT ((0.B = 'A') OR (0.M = 'N') OR (0.M = 'N')) ! <== incorrect
  Get     Retrieval by index of relation 0:A
    Index name  PZY_X [1:1]
      Keys: 0.P = 'Q'
 T      M          P        B
 S      M          Q        B
 S      M          Q        NULL
 S      N          P        B
 S      N          P        NULL
 S      N          Q        B
 S      N          Q        NULL
 S      N          NULL     B
 S      N          NULL     NULL
 S      NULL       Q        B
 S      NULL       Q        NULL
10 rows selected
```

The sequential access strategy gives the correct result as seen in the following example.

```
select t,m,p,b from a
  where (t='S' and (m='N' or p='Q')) or (t='Z' and (m='N' or b='A'))
  order by t,m,p,b optimize for sequential access;
 T      M          P        B
 S      M          Q        A              <= missing row
 S      M          Q        B
 S      M          Q        NULL
 S      N          P        A              <= missing row
 S      N          P        B
 S      N          P        NULL
 S      N          Q        A              <= missing row
 S      N          Q        B
 S      N          Q        NULL
 S      N          NULL     A              <= missing row
 S      N          NULL     B
 S      N          NULL     NULL
 S      NULL       Q        A              <= missing row
 S      NULL       Q        B
 S      NULL       Q        NULL
15 rows selected
```

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

1. The main select query contains a where clause with range list OR predicates that involves 4 columns, each testing equality with a constant literal value. In this example, we use the column names B, M, P, and T.
2. The column T is a common segment between index BTY_X and MTZ_X, where BTY_X is an index on columns B, T and Y; MTZ_X is an index on columns M, T, and Z. The column P is defined as a leading segment in PZY_X.
3. The main OR predicate has the left branch which contains an AND between "T='S'" and another secondary OR predicate "(m='N' or p='Q')". The right branch contains an AND between "T='Z'" and another secondary OR predicate "(m='N' or b='A')".
4. The OR predicates are arranged in such a way so that the strategy of the optimizer uses the range list retrieval "MTZ_X [(2:2)2]" on keys "r0: (0.M = 'N') AND (0.T = 'S')" and "r1: (0.M = 'N') AND (0.T = 'Z')" in the second leg of the "inner" OR index retrieval under the first leg of the "outer" OR index retrieval.
5. The NOT filter, created at the top of the second leg of the "inner" OR index retrieval, DOES NOT contain the equality predicate "0.T = 'Z'" from the first leg.
6. The NOT filter, created at the top of the second leg of the "outer" OR index retrieval, DOES NOT contain the predicates "(0.T = 'S')" and "(0.T = 'Z')" from the range list predicates of the first leg.

There is no known workaround for this problem.

This problem has been corrected in Oracle Rdb Release 7.0.6.3.

3.1.3 Performance Problems when RDM$BIND_SNAP_QUIET_POINT Defined to 0

Bug 884004

When the logical name RDM$BIND_SNAP_QUIET_POINT was defined to 0, it would cause Oracle Rdb to write out modified buffers and demote all page buffer locks when a READ ONLY transaction was started. This would defeat the optimizations utilized by the FAST COMMIT feature, and would also cause additional locking and page buffer I/O.

This problem has been corrected in Oracle Rdb Release 7.0.6.3. When the RDM$BIND_SNAP_QUIET_POINT logical is defined to 0 and a process is holding the quiet point lock when starting a READ ONLY transaction, the quiet lock will be retained. Thus buffers will not be flushed and page locks will not be released when starting a READ ONLY transaction. If a backup process requests the quiet point lock, and the logical RDM$BIND_SNAP_QUIET_POINT is defined to 0, then any READ ONLY transactions will immediately write out modified buffers and release the quiet point lock.

3.1.4 Workload Ignored When Loaded With RMU/INSERT OPTIMIZER_STATISTICS

In previous versions of Oracle Rdb, if workload statistics were loaded into a database using the *RMU/INSERT OPTIMIZER_STATISTICS* command, the workload would be ignored by the optimizer.

The use of workload statistics can be observed by setting the *ESTIMATES* debug flag as shown in the following example.

```
SQL> set flags 'estimates';
SQL> select * from t1 where f1=1;
Solutions tried 1
Solutions blocks created 1
Created solutions pruned 0
Cost of the chosen solution   3.0000000E+00
Cardinality of chosen solution   1.0000000E+00
~O: Workload statistics used
          F1            F2
           1             1
1 row selected
```

After loading workload statistics with the *RMU/INSERT* command, a query that should use statistics will fail to show the *~O: Workload statistics used* message. This indicates that the statistics are being ignored.

The problem can be identified by examining the data loaded into the *RDB$WORKLOAD* system table. If the *RDB$CREATED* and *RDB$LAST_ALTERED* columns have the same value, as shown in the following example, then workload statistics will be ignored.

```
SQL> select rdb$created,rdb$last_altered from rdb$workload;
 RDB$CREATED               RDB$LAST_ALTERED
 19-OCT-2001 00:33:53.27   19-OCT-2001 00:33:53.27
1 row selected
```

The problem can be corrected by manually updating the *RDB$LAST_ALTERED* column, as shown in the following example. New attaches will commence using the workload values.

```
SQL> update rdb$workload set rdb$last_altered=current_timestamp;
```

This problem was actually corrected in Oracle Rdb Release 7.0.6.2 but the release note was inadvertently omitted.

3.1.5 Zero Index Prefix Cardinality After Create Index

Bug 867890

Under certain conditions, index prefix cardinality stored for a newly−created sorted index was incorrect (zero). This could sometimes occur when a table already had rows stored in it. When the index prefix cardinalities are not stored (are zero), the query optimizer might choose poor query strategies resulting in slow response times.

The following is an example illustrating the problem. A table, TT, is created with two data rows. Next, a unique index, TT_U, is created on that table and the transaction is committed. The ensuing select statement lists the index segments and the index prefix cardinality stored for each segment. For index TT_U, which has three segments, there are two index prefixes: (1) the column S by itself, and (2) the column S with the column E. The example below shows that the index prefix cardinalities were zero both after the index creation was committed and also after a disconnect from the database had been performed.

```
SQL> create table tt (s char (4), e char (1), v int);
SQL> insert into tt values ('ABC', 'Z', 10000000);
1 row inserted
SQL> insert into tt values ('ABC', 'Z', 10000001);
1 row inserted
SQL> commit;
SQL>
SQL> create unique index tt_u on tt (s,e,v);
SQL> commit;
SQL>
SQL> select cast(rdb$field_name as char(1)) as col,
cont>        cast(rdb$field_position as tinyint) as pos,
cont>        cast(rdb$cardinality as tinyint) as pfx_card
cont>   from rdb$index_segments where rdb$index_name = 'TT_U';
 COL     POS   PFX_CARD
 S         1          0
 E         2          0
 V         3          0
3 rows selected
SQL> rollback;
SQL>
SQL> disconnect all;
SQL>
SQL> attach 'filename test.rdb';
SQL>
SQL> select cast(rdb$field_name as char(1)) as col,
cont>        cast(rdb$field_position as tinyint) as pos,
cont>        cast(rdb$cardinality as tinyint) as pfx_card
cont>   from rdb$index_segments where rdb$index_name = 'TT_U';
 COL     POS   PFX_CARD
 S         1          0
 E         2          0
 V         3          0
3 rows selected
SQL> rollback;
```

As a workaround, to correct this error following index creation, use the RMU utility to collect optimizer cardinality statistics for the problem index.

```
$ rmu/collect optimizer /statistic=cardinality test.rdb
```

This problem has been corrected in Oracle Rdb Release 7.0.6.3. Now, index prefix cardinalities will be recorded for newly−created indexes as soon as the work is committed.

3.1.6 RDB−E−ARITH_EXCEPT Error From the Rdb Optimizer

Bug 1694309

When using workload statistics, it was possible that a query that joined several tables together would produce a divide by zero error.

The following example shows the result of trying to execute a query that exposed the problem.

```
%RDB−E−ARITH_EXCEPT, truncation of a numeric value at runtime
−SYSTEM−F−HPARITH, high performance arithmetic trap, Imask=00000000,
  Fmask=00000001, summary=04, PC=0000000000FBF748, PS=0000000B
−SYSTEM−F−FLTDIV, arithmetic trap, floating/decimal divide by zero at
  PC=0000000000FBF748, PS=0000000B
```

As a side effect of this problem, some queries could be inaccurately costed by the optimizer, which could lead to less than optimal retrieval strategies. The following simple example shows a query where the cardinality was inaccurately calculated from the workload statistics because of this problem.

```
SQL> set flags 'estimates'
SQL>  select * from t1, t2 where t1.f1=t2.f1;
Solutions tried 6
Solutions blocks created 4
Created solutions pruned 1
Cost of the chosen solution   1.5162601E+01
Cardinality of chosen solution   0.0000000E+00
~O: Workload statistics used
       T1.F1         T2.F1
           1             1
.
.
.
1000 rows selected
```

Oracle Rdb now correctly interprets NULL factors of 1.0 and 0.0 in workload statistics and therefore correctly calculates the cardinality of this example to 1000 rows.

The problem can be worked around using any of the following techniques:

- Ensuring that workload data does not have a null factor of exactly 0.0 or 1.0.
- Removing workload statistics.
- Ensuring that the table cardinalities are greater than 1 for all tables in the query.
- Use of the *OLD_COST_MODEL* debug flag.

This problem has been corrected in Oracle Rdb Release 7.0.6.3.

3.1.7 COMPUTED BY Columns Now Automatically Reserve Referenced Tables

Bug 1253235

In previous versions of Oracle Rdb, it was possible that an application could fail if a reference to a COMPUTED BY or view column required a table not specified in the RESERVING clause of the SET or DECLARE TRANSACTION statement.

The application developer may not know that a column requires these extra tables as part of the transaction, or the definition of the view or COMPUTED BY column may be changed to reference different tables after the application is in production.

The following code shows an example where a COMPUTED BY column (PRICE) requires access to a table (CASE_TABLE) that was not referenced by the RESERVING clause.

```
SQL> set transaction read only
cont>     reserving REPORT_VIEW for shared read;
SQL> select * from REPORT_VIEW order by LINE_NUM;
%RDB-E-UNRES_REL, relation CASE_TABLE in specified request is not a
relation reserved in specified transaction
SQL> rollback;
SQL> set transaction read only
cont>     reserving REPORT_VIEW, CASE_TABLE for shared read;
SQL> select * from REPORT_VIEW order by LINE_NUM;
 CASE_NUM              LINE_NUM                     PRICE
 1                            1                   7270.00
 1                            2                  14540.00
2 rows selected
```

This problem has been corrected in Oracle Rdb Release 7.0.6.3. Rdb now automatically reserves tables referenced by COMPUTED BY columns for SHARED READ.

3.1.8 Bugchecks in PIOGB$PURGE_BUFFER After Node Failure When Row Cache in Use

Bug 2058891

When the Row Cache feature was enabled with global buffers, it was possible for processes to bugcheck with the following exception after a node failure occurred:

```
***** Exception at 00E58F9C : PIOGB$PURGE_BUFFER + 0000078C
%COSI-F-BUGCHECK, internal consistency failure
```

The problem could also occur the first time the database was accessed after an RMU/CLOSE/ABORT=DELPRC command was issued.

There was a problem in the database recovery mechanisms for the Row Cache feature that could cause global buffer data structures to become inconsistent.

This problem has been corrected in Oracle Rdb Release 7.0.6.3.

3.1.9 Page Locking Problems in Release 7.0.6.2

Bugs 2042873 and 2061266

Oracle Rdb Release 7.0.6.2 introduced errors into the buffer page locking mechanisms that could cause excessive stalls or deadlocks.

The first problem was triggered when the Asynchronous Prefetch (APF) mechanism was used to fetch a buffer that contained only one page. In that situation, blocking ASTs for the page lock would be ignored. This was typically seen for buffers containing Space Area Management (SPAM) pages.

Regular user processes rarely read SPAM pages via APF, but the AIJ Log Recovery Server (LRS) will often use APF to read SPAM pages. Processes attempting to read the standby database while the LRS was in operation would sometimes see long stalls for SPAM page locks since the LRS was neglecting to process the blocking AST requests.

When not using Hot Standby, this problem may be avoided by disabling APF. However, it is not possible to disable APF for the LRS.

The second problem was seen when Global Buffers were enabled. In that situation, if one process read a buffer via the APF mechanism, and a second process wanted to access pages within the same buffer, the second process would not use the proper locking protocol to ensure that the first process was properly notified via the blocking AST mechanisms. This could lead to excessive stalls for page locks and deadlocks on page locks. This problem was quite noticeable when the LRS process needed to access a page being held by processes doing online access to the standby database. It was possible for the LRS to encounter so many lock conflicts that it could not process fast enough and would throttle activity on the master database.

To workaround this problem, global buffers may be disabled. This may, however, induce a substantial performance degradation in the application.

These problems have been corrected in Oracle Rdb Release 7.0.6.3.

3.1.10 Poor Choice of Indexes by Dynamic Optimizer

Bug 703558

A query that worked well in Rdb6 took ten times longer to execute in Rdb 7.0. The problem was attributed to a poor choice of indexes used by the dynamic optimizer. Here is the query:

```
select a.ass_asset_code, a.ass_asset_name, i.tot_clients, i.tot_value
  from (select ass_asset_code,
               count (*) as tot_clients,
               sum(asset_value) as tot_value
        from investment
        where dlr_dealer_id starting with ''                    <-- note
          and ofc_office_id starting with '0119027BO01'         <-- note
          and adv_adviser_id starting with ''                   <-- note
          and cln_service_type starting with ''                 <-- note
        group by ass_asset_code) i,
       asset a
  where i.ass_asset_code = a.ass_asset_code
  order by a.ass_asset_code asc;
```

The WHERE clause includes these conditions:

```
    dlr_dealer_id starting with ''
```

```
    ofc_office_id starting with '0119027BO01'
    adv_adviser_id starting with ''
    cln_service_type starting with ''
```

The Rdb6 strategy chosen was the following:

```
Conjunct
Match
  Outer loop
    Merge of 1 entries
      Merge block entry 1
      Aggregate       Sort
      Leaf#01 BgrOnly INVESTMENT Card=383229
        BgrNdx1 INVESTMENT_NDX_7 [1:1] Fan=14                  <-- note
        BgrNdx2 INVESTMENT_NDX_6 [1:1] Fan=14                  <-- note
        BgrNdx3 INVESTMENT_NDX_5 [1:1] Fan=14                  <-- note
        BgrNdx4 INVESTMENT_NDX_3 [1:1] Bool Fan=7              <-- note
  Inner loop      (zig-zag)
    Get     Retrieval by index of relation ASSET
      Index name  ASSET_NDX_2 [0:0]
```

Use of four background indexes makes sense because each has a different leading segment (column) matching one of the STARTING WITH clauses. The execution trace (not shown) indicates that the background scanned BgrNdx2 (INVESTMENT_NDX_6) to completion, but aborted all other scans due to reaching FtchLim. This also makes sense because the leading segment of this index is OFC_OFFICE_ID, which is the only column for which a real value is provided in the STARTING WITH clause. In other words, Rdb is able to retrieve the necessary rows using index INVESTMENT_NDX_6 without having to do a full index scan.

The Rdb 7.0 strategy chosen was the following:

```
Conjunct
Match
  Outer loop
    Merge of 1 entries
      Merge block entry 1
      Aggregate       Sort
      Leaf#01 BgrOnly INVESTMENT Card=383229
        BgrNdx1 INVESTMENT_NDX_3 [1:1] Bool Fan=7              <-- note
        BgrNdx2 INVESTMENT_NDX_1 [1:1] Bool Fan=5              <-- note
  Inner loop      (zig-zag)
    Get     Retrieval by index of relation ASSET
      Index name  ASSET_NDX_2 [0:0]
```

Note that INVESTMENT_NDX_6 was not selected as a candidate index. This means that whichever index is chosen, a full index scan will have to be performed since the STARTING WITH clauses on these indexes have values of an empty string ('). The end result is that there is an order of magnitude more I/O for Rdb 7.0.

As a workaround, a query outline can be defined. However, in Oracle Rdb Release 7.0.1.2, the version under which the problem was reported, it was not possible to work around the problem by defining a query outline. That was a separate problem. A correction to allow a query outline to be used in this case became available in Oracle Rdb Release 7.0.2.

This problem has been corrected in Oracle Rdb Release 7.0.6.3.

3.1.11 Storage Area Default Size Increase

Bug 2151253

The storage area default size was 400 pages which was too small and always caused the area to be extended at least once during database creation. This default has been increased to 600 pages which is now just large enough to not require extending during database creation.

This problem has been corrected in Oracle Rdb Release 7.0.6.3.

3.1.12 Query Slows Down Using Full Index Scan [0:0]

Bug 1635351

A query that worked well in Oracle Rdb Release 7.0.1.2 became much slower in Oracle Rdb Release 7.0.6 when using full index scan. Even if the customer uses the same outline as before, the performance does not improve. Here is the query:

```
select h.hnmei_id,
       h.hnmei_nm
from   pm_zumen_v p,
       zumen_v    z,
       hinmei_v   h
where  p.hinban = '000704419' and
       p.zuban  = z.zuban and
       z.teisei_kgo in ( select max(z1.teisei_kgo)
                           from zumen_v z1
                           where z.zuban = z1.zuban ) and
       z.zuban      = h.zuban and
       z.teisei_kgo = h.teisei_kgo
```

The Oracle Rdb Release 7.0.1.2 strategy chosen was the following:

```
Cross block of 4 entries
  Cross block entry 1
    Index only retrieval of relation PM_ZUMEN
      Index name  IDX_PM_ZUMEN_0 [1:1]
  Cross block entry 2
    Conjunct        Index only retrieval of relation ZUMEN
      Index name  IDX_ZUMEN_0 [1:1]
  Cross block entry 3
    Conjunct        Aggregate      Index only retrieval of relation ZUMEN
      Index name  IDX_ZUMEN_0 [2:2]  Min key lookup
  Cross block entry 4
    Index only retrieval of relation HINMEI
      Index name  IDX_HINMEI_0 [3:3]
0 rows selected
```

The Oracle Rdb Release 7.0.6 strategy chosen was the following:

```
Cross block of 3 entries
  Cross block entry 1
    Conjunct
    Match
      Outer loop
        Index only retrieval of relation ZUMEN
          Index name  IDX_ZUMEN_0 [0:0]          <-- full index scan
        Inner loop      (zig-zag)
          Aggregate      Index only retrieval of relation ZUMEN
            Index name  IDX_ZUMEN_0 [0:0]        <-- full index scan
  Cross block entry 2
    Conjunct        Index only retrieval of relation PM_ZUMEN
      Index name  IDX_PM_ZUMEN_0 [1:1]
  Cross block entry 3
```

```
    Index only retrieval of relation HINMEI
      Index name  IDX_HINMEI_0 [3:3]
0 rows selected
```

There is no workaround available for this problem. Even an outline that switches from match to cross strategy is unable to apply full index scan.

This problem has been corrected in Oracle Rdb Release 7.0.6.3.

3.1.13 Recovery Process Caused Excessive Snapshot File Growth

Bug 2033576

In Oracle Rdb Release 7.0.6.2, it was possible for the Database Recovery process (DBR) to excessively extend snapshot files and perhaps fail with a bugcheck dump containing an error similar to the following:

```
***** Exception at 0017040C : PIO$EXTEND_STAREA + 0000097C
%RDMS-F-FILACCERR, error extending file DEV:[DIR]SNAPSHOT_FILE.SNP;
-SYSTEM-W-DEVICEFULL, device full; allocation failure
```

This would typically happen after a process had inserted many rows in the database and, before the transaction was committed, there was a system failure; or the database was closed with the RMU/CLOSE/ABORT=DELPRC command. In that situation, the DBR would needlessly store before−image entries of all of the inserted rows into the snapshot file(s) and it would not attempt to reuse any of the pages currently in the snapshot file(s).

This problem has been corrected in Oracle Rdb Release 7.0.6.3. After a node failure, the DBR will not attempt to write snapshot file entries when rolling back inserted rows.

3.2 SQL Errors Fixed

3.2.1 Select With Identical "not in" Clauses Causes Bugcheck

Bug 1978741

A SQL query which contains two identical "not in" clauses can cause an application to crash, terminate or bugcheck.

The following example shows a SQL statement that will cause the error:

```
select count(*) from JOBS
 where JOB_CODE not in ('A', 'B')
   and JOB_CODE not in ('A', 'B');
```

As a workaround for this problem, remove all duplicate "not in" clauses.

This problem has been corrected in Oracle Rdb Release 7.0.6.3.

3.2.2 Queries Ending in Reserved Words Fail to Execute in Dynamic SQL

Bug 2088594

If the final token of a query is a column whose name is a reserved word, then the query may fail with SQL−F−PREMATURE_EOF. However, if extra syntax is added to the query it will work. Similarly, if the column is prefixed with the table name or correlation name (such as TT.POSITION), then the query succeeds.

The following example shows the problem using a dynamic SQL program. When the query is extended by adding an additional column to the ORDER BY clause the query succeeds.

```
>> CREATE TABLE TT (AA INT, POSITION INT)
>> INSERT INTO TT (AA, POSITION) VALUES (1, 1)
>> INSERT INTO TT (AA, POSITION) VALUES (1, 2)
>> SELECT * FROM TT ORDER BY POSITION
error: −1...
error text:
%SQL−F−PREMATURE_EOF, Statement is syntactically incomplete
>> SELECT * FROM TT ORDER BY POSITION, AA
out: 0:        0
out: 1:        0
 0/AA: INTEGER:1
 1/POSITION: INTEGER:1
 0/AA: INTEGER:1
 1/POSITION: INTEGER:2
>> ROLLBACK
```

The problem in this case is that POSITION is valid starting syntax for the POSITION function. Dynamic SQL requests the next token which is expected to be the start of the function argument list. However, an exception is raised because dynamic SQL does not permit continuations of statements. Similar problems occur if column names such as TRIM and SUBSTRING are used.

If this query were executed by interactive SQL, then the terminating semicolon (;) would indicate that the builtin function was not being used and the name is then treated as a column name.

To solve this problem, the next release of dynamic SQL will permit an optional terminating semicolon (;). If more tokens are requested (as in this problem case) an implicit ; will be provided by SQL and the failing syntax may succeed.

This problem has been corrected in Oracle Rdb Release 7.0.6.3.

3.2.3 SQL$MOD Compiler Does Not Recognize G_FLOAT With COBOL

Bug 1149572

COBOL on VAX only supported D_FLOAT and not G_FLOAT. G_FLOAT support was added on Alpha but SQL$MOD still gave a warning for it. For example, suppose a SQL Module Language program for the COBOL language declared a procedure with a parameter called ":P_FLOATFLD" which is of type "FLOAT". In this case, if the program was compiled with a /G_FLOAT qualifier, SQL$MOD would flag the declaration as having an unsupported datatype as follows:

```
$SQL$MOD/G_FLOAT EXAMPLE_PROG.SQLMOD
        :P_FLOATFLD     FLOAT);
                        1
%SQL-W-LANUNSDTP, (1) COBOL does not support the data type for parameter
P_FLOATFLD
```

This program will now compile without warnings on OpenVMS Alpha. The warning still (appropriately) appears for OpenVMS VAX.

This problem has been corrected in Oracle Rdb Release 7.0.6.3.

3.3 Oracle RMU Errors Fixed

3.3.1 RMU/ANALYZE/CARDINALITY Fails on Databases with Local Temporary Tables

Bug 2019322

RMU/Analyze/Cardinality, when attempting to process LOCAL temporary tables, generated an error and failed to execute.

```
$ rmu/analyze/cardinality sql$database
%RDMS-E-BAD_CODE, corruption in the query string
%RMU-F-FATALRDB, Fatal error while accessing Oracle Rdb.
%RMU-F-FTL_ANA, Fatal error for ANALYZE operation at 27-SEP-2001 13:34:25.79
```

RMU has now been corrected to ignore temporary tables as well as views. The workaround for this problem is to use the RMU/SHOW OPTIMIZER/STATISTIC=CARD command, or the RMU/COLLECT OPTIMIZER_STATISTICS command if RMU/ANALYZE/CARDINALITY/UPDATE was tried.

This problem has been corrected in Oracle Rdb Release 7.0.6.3.

3.3.2 RMU/COPY/BLOCKS_PER_PAGE Can Corrupt Copied Database

Bug 2028181

For the RMU/COPY command, if the "/blocks_per_page" qualifier was not specified for a particular storage area but for all database storage areas, database corruption of uniform storage areas occurred to the copied database. As documented in the Oracle Rdb RMU Reference Manual for the RMU/COPY command, BLOCKS PER PAGE can only be changed for MIXED storage areas, not UNIFORM storage areas. But when the "/blocks_per_page" qualifier was used for all storage areas, RMU incorrectly bypassed the check for UNIFORM storage areas and attempted to change the BLOCKS PER PAGE setting for UNIFORM as well as MIXED storage areas. This caused the database corruption of the moved copy of the database. Now the number of BLOCKS PER PAGE will be changed only for MIXED storage areas and a warning message will be output for each UNIFORM storage area that BLOCKS PER PAGE cannot be changed for that area since it is a UNIFORM database storage area.

The following example shows that since /BLOCKS_PER_PAGE=3 was specified for all storage areas in the mf_personnel database, it caused the database corruption problem for the uniform storage areas in the copied database.

```
$ RMU/COPY/DIR=TMPDIR/ROOT=TMPDIR:MFP1 /NOLOG /BLOCKS_PER_PAGE=3 MF_PERSONNEL
%RMU-W-BADPTLARE, invalid larea for uniform data page 5 in storage area 1
%RMU-W-BADPTLAR2,                   SPAM larea_dbid: 16385, page larea_dbid: 1
%RMU-W-BADPTLARE, invalid larea for uniform data page 149 in storage area 1

$  RMU/VERIFY/ALL TMPDIR:MFP1
%RMU-I-BGNROOVER, beginning root verification
%RMU-I-ENDROOVER, completed root verification
%RDB-W-NO_RECORD, access by dbkey failed because dbkey is no longer associated
 with a record
-RDMS-F-NODBK, 61:1179:0 does not point to a data record
%RMU-E-ERRRDBREL, error accessing RDB$RELATIONS relation
```

The following example shows that the problem is now fixed.

```
$ RMU/COPY/DIR=TMPDIR/ROOT=TMPDIR:MFP1 /NOLOG /BLOCKS_PER_PAGE=3 MF_PERSONNEL
%RMU-W-UNIFORMBLOCKS, BLOCKS PER PAGE cannot be changed for uniform storage
 area RDB$SYSTEM
%RMU-W-UNIFORMBLOCKS, BLOCKS PER PAGE cannot be changed for uniform storage
area MF_PERS_SEGSTR

$  RMU/VERIFY/ALL TMPDIR:MFP1
$
```

To avoid this problem, specify /BLOCKS_PER_PAGE for each individual storage area in the RMU/COPY command, not as a default for all storage areas.

This problem has been corrected in Oracle Rdb Release 7.0.6.3.

3.3.3 DROPped Storage Area and RMU/VERIFY in Cluster

Bug 1421362

Previously, when a database was opened in a cluster environment, it was possible for the RMU/VERIFY command to be unable to open storage area files when storage areas were moved or dropped on another node in the cluster.

For example, consider the following sequence of events on a two node cluster (consisting of NODE1 and NODE2):

```
Node1$: RMU /OPEN MFP

Node2$: RMU /OPEN MFP

Node1$: SQL$ ALTER DATABASE FILENAME MFP DROP STORAGE AREA U1;

Node2$: RMU /VERIFY MFP
    .
    .
    .
%RMU-F-OPNFILERR, error opening file U1.RDA
%RMU-F-FILNOTFND, file not found
%RMU-E-BDAREAOPN, unable to open file U1.RDA  for storage area
%RMU-F-ABORTVER,  fatal error encountered; aborting verification
```

This problem has been corrected in Oracle Rdb Release 7.0.6.3. The RMU/VERIFY utility now correctly detects storage areas that have been dropped or moved.

3.3.4 RMU Fails to Perform OPTIMIZER_STATISTICS Actions on Some Databases

In prior versions of Oracle Rdb, attempts to use RMU/SHOW OPTIMIZER_STATISTICS, RMU/COLLECT OPTIMIZER_STATISTICS, and related commands would fail if the default database character set was not DEC_MCS.

The following example shows the problem for a DEC_KANJI database.

```
$ rmu/show optimizer_statistics DISK1:[TESTING]SAMPLE.RDB
%RDB-F-CONVERT_ERROR, invalid or unsupported data conversion
-RDMS-E-CSETBADCOMPARE, incompatible character sets prohibit the requested
```

```
comparison
%RMU-F-FATALRDB, Fatal error while accessing Oracle Rdb.
%RMU-F-FTL_SHOW, Fatal error for SHOW operation at 29-OCT-2001 16:31:20.59
$
$ rmu/collect optimizer_statistics DISK1:[TESTING]SAMPLE.RDB
%RDB-F-CONVERT_ERROR, invalid or unsupported data conversion
-RDMS-E-CSETBADCOMPARE, incompatible character sets prohibit the requested
comparison
%RMU-F-FATALRDB, Fatal error while accessing Oracle Rdb.
%RMU-F-FTL_ANA, Fatal error for ANALYZE operation at 29-OCT-2001 16:31:36.12
```

This problem has been corrected in Oracle Rdb Release 7.0.6.3.

3.3.5 RMU Tape Density Problems Starting With VMS V7.2−1

Bugs 1362656 and 1432269

Starting with Compaq VMS V7.2−1, there were density problems for RMU commands that allow tape density values to be specified with the /DENSITY qualifier: RMU/BACKUP, RMU/BACKUP/AFTER_JOURNAL and RMU/OPTIMIZE_AIJ. These problems resulted in one of the following tape density related errors being returned when density values which were correct were specified. These values worked when specified in RMU commands prior to VMS V7.2−1. The problems occurred with tape cartridges initialized to the new VMS V7.2−1 MTD compaction values.

```
%RMU-E-DENSITY, TAPE_DEVICE:[000000]DATABASE.BCK; does not support specified
  density
%RMU-E-POSITERR, error positioning TAPE_DEVICE:
```

These problems resulted from problems in VMS tape device drivers which were enhanced to handle the new MTD (multiple tape density) values introduced in VMS V7.2−1. These problems caused the device drivers to incorrectly handle the existing tape density codes used prior to VMS V7.2−1. These problems exist in VMS (some have been corrected) and cannot be fixed by RMU. But RMU has been changed to avoid this problem by allowing the new MTD density codes to be specified by the /DENSITY command using the following syntax.

```
/DENSITY=(new_density_value,[NO]COMPACTION)
```

The existing density values can continue to be specified using the same syntax as before.

```
/DENSITY=existing_density_value
```

Please see the New Feature documentation on this enhancement for a full description (Section 6.2.6).

The following example shows the error returned when a valid density code was specified for a tape device with VMS V7.2−1.

```
$RMU/BACKUP/DENSITY=70000/REWIND/LABEL=(LABEL1,LABEL2) MF_PERSONNEL
TAPE1:MFP.BCK, TAPE2:
%RMU-E-POSITERR, error positioning TAPE1:
```

This problem could sometimes be avoided by initializing the tape with VMS V7.2−1 commands and not setting the density in the RMU command.

This problem has been corrected in Oracle Rdb Release 7.0.6.3.

3.4 Hot Standby Errors Fixed

3.4.1 Oracle Rdb Release 7.0.6.2 Process Hangs During AIJ Switchover

In Oracle Rdb Release 7.0.6.2, it was possible to encounter hang problems when using the Hot Standby feature if user processes on the master database had multiple database attaches. This problem was introduced in Release 7.0.6.2.

If a process was attached to multiple databases and the AIJ Log Server (ALS) process was enabled, it was possible for processes to hang with the stall message "hibernating on AIJ submission". One process usually was hung with the stall message "waiting for RTUPB list (EX)". The only way to resolve the problem was to terminate the process that was hanging with "waiting for RTUPB list (EX)".

This problem has been corrected in Oracle Rdb Release 7.0.6.3.

3.5 RMU Show Statistics Errors Fixed

3.5.1 Stream ID Format is Different in Different Places

Bug 2093770

The Stream ID display has been made uniform everywhere it appears.

This problem has been corrected in Oracle Rdb Release 7.0.6.3.

3.5.2 AUTO_RECONNECT Variable Value is not Honored When Imported From a RMU/SHOW STATISTICS Configuration File

Bug 2113645

The AUTO_RECONNECT parameter value was not honored when imported from a RMU/SHOW STATISTICS configuration file.

This problem has been corrected in Oracle Rdb Release 7.0.6.3.

3.5.3 Some RMU/SHOW STATISTICS Counters Can Be Used To Define Events In Interactive Mode But Not In Batch Mode

Bug 2078940

Some RMU/SHOW STATISTICS counters such as "−prom−deadlocks" can be used to define events in interactive mode but not in batch mode.

This problem has been corrected in Oracle Rdb Release 7.0.6.3.

3.5.4 RMU/SHOW STATISTICS Online Analysis Configuration Options Do Not Work Properly

Bug 1893049

RMU/SHOW STATISTICS online analysis configuration options did not use the right percentile for displaying read−write and read−only statistics.

This problem has been corrected in Oracle Rdb Release 7.0.6.3.

3.5.5 Missing "U" for Utility Jobs in RMU/SHOW STATISTICS Displays

Bug 2110027

A "U" was not displayed for utility jobs in RMU/SHOW STATISTICS displays.

This problem has been corrected in Oracle Rdb Release 7.0.6.3.

3.5.6 RMU/SHOW STATISTICS Mixes Up Count Labels

Bug 1937577

In the RMU/SHOW STATISTICS utility, the count labels associated with row cache search are mixed up.

This problem has been corrected in Oracle Rdb Release 7.0.6.3.

3.5.7 Errors in Saved RMU/SHOW STATISTICS Configuration File

Bug 1922670

There are three errors in the saved RMU/SHOW STATISTICS configuration file.

- The RUJ_FILE_SIZE parameter is documented to default to 256 but is saved as 25.6 in the configuration file.
- If you are monitoring more than one node and save the configuration file, the current node name is not correctly saved.
- If monitoring more than one node, the CLUSTER_NODES parameter is saved with trailing garbage characters.

These problems have been corrected in Oracle Rdb Release 7.0.6.3.

3.5.8 RMU/SHOW STATISTICS Shows Incorrect Area Sizes

Bug 2151237

The RMU/SHOW STATISTICS display of storage area information shows the initial page count statistic two times. Further, the count displayed is not accurate.

This problem has been corrected in Oracle Rdb Release 7.0.6.3. The accurate page count is now displayed only once.

3.5.9 RMU/SHOW STATISTICS Allowed Suspend of Disabled ABS

Previously, the RMU /SHOW STATISTICS utility allowed the user to suspend AIJ Backup Server (ABS) operations on a node even when the ABS was disabled. This could lead to confusing errors during later manual AIJ backup operations.

This problem has been corrected in Oracle Rdb Release 7.0.6.3. The RMU /SHOW STATISTICS utility now does not allow the ABS to be suspended when it is not enabled.

3.5.10 Area Locks Demoted Statistic Not Always Correctly Incremented

Previously, the "locks demoted" statistic for "area" locks was not always correctly incremented. This could occur, for example, when a read−only transaction started when the previous transaction was a read−write transaction. The "locks promoted" statistic could have been incorrectly incremented in this case. This, in turn, lead to potentially confusing results when comparing the "locks promoted" rate with the "locks demoted" rate

for "area" locks in the "RMU/SHOW STATISTICS" facility.

This problem has been corrected in Oracle Rdb Release 7.0.6.3. The correct statistic is now incremented when an "area" lock is demoted from one lock mode to a lower mode.

3.5.11 RMU/SHOW STATISTICS Does Not Honor CHECKPOINT_SORT

Bug 2057091

There was a problem wherein the CHECKPOINT_SORT in the RMU/SHOW STATISTICS configuration file was not being honored.

This problem has been corrected in Oracle Rdb Release 7.0.6.3.

3.5.12 RMU/SHOW STATISTICS CHECKPOINT_ALARM Does Not Give Out OPCOMs

Bug 1735654

The CHECKPOINT_ALARM variable is no longer used to give out operator notification messages (OPCOM) for long transactions. The variable LONG_TX_SECONDS is now used for this purpose. RMU/SHOW STATISTICS gives out OPCOMs to indicate transactions that exceed the interval specified by the LONG_TX_SECONDS at intervals of 1 minute. The OPCOMs are delivered to the OPCOM classes specified by the NOTIFY variable in the configuration file.

This problem has been corrected in Oracle Rdb Release 7.0.6.3.

Chapter 4
Software Errors Fixed in Oracle Rdb Release 7.0.6.2

This chapter describes software errors that are fixed by Oracle Rdb Release 7.0.6.2.

4.1 Software Errors Fixed That Apply to All Interfaces

4.1.1 Query with UNION Subselect Returns Wrong Results

Bug 1656974

The following query with UNION subselect should return 0 rows.

```
set flags 'strategy,detail';
select ps.id, ps.kbn, ps.ymd
 from (select ps1.id,
              ps1.kbn,
              '99999999'                    ! <== this causes the problem
         from  ps ps1, pm pm
         where pm.id = ps1.id
       union all
       select ps2.id,
              ps2.kbn,
              ps2.end_ymd
         from  ps ps2, pm pm
         where pm.id = ps2.id)

      as ps (id, kbn, ymd)
  where ps.id = '021023307' and
        ps.ymd > '12345678' and
        ps.kbn in ('1','2') ;
Tables:
  0 = PS
  1 = PM
  2 = PS
  3 = PM
Merge of 1 entries
  Merge block entry 1
  Merge of 2 entries
    Merge block entry 1
    Conjunct: 1.id = 0.ID
    Match
      Outer loop      (zig-zag)
        Conjunct: 0.ID = '021023307'
        Conjunct: '99999999' > '12345678'
        Get     Retrieval by index of relation 0:PS
          Index name  IDX_PS_2 [1:1] Bool
            Key: <mapped field> = '021023307'
            Bool: '99999999' > '12345678'
      Inner loop      (zig-zag)
        Index only retrieval of relation 1:PM
          Index name  IDX_PM_0 [0:0]
    Merge block entry 2
    Conjunct: 3.id = 2.ID
    Match
      Outer loop      (zig-zag)
        Conjunct: (2.ID = '021023307') AND (2.end_ymd > '12345678')
                   AND ((2.kbn = '1') OR (2.kbn = '2'))
        Get     Retrieval by index of relation 2:PS
          Index name  IDX_PS_2 [2:1]
            Key: (<mapped field> = '021023307') AND (<mapped field> > '12345678'
                 )
      Inner loop      (zig-zag)
        Index only retrieval of relation 3:PM
          Index name  IDX_PM_0 [0:0]
 ID          KBN   YMD
```

```
 021023307   0     99999999
1 row selected
```

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

1. The query contains a subselect of a UNION, where one of the columns is a literal, e.g. '99999999'.
2. The where clause contains an equality predicate, a GTR predicate, and an IN clause.

As a workaround, the query works if the IN clause is moved before the GTR predicate, as in the following example.

```
set flags 'strategy,detail';
! The following query should return 0 rows
!
select ps.ID, ps.kbn, ps.ymd
 from (select ps1.ID,
              ps1.kbn,
              '99999999'
         from  ps ps1, pm pm
         where pm.id = ps1.ID
       union all
       select ps2.id,
              ps2.kbn,
              ps2.end_ymd
         from  ps ps2, pm pm
         where pm.id = ps2.id)

      as ps (id, kbn, ymd)
  where ps.id = '021023307' and
        ps.kbn in ('1','2') and         <=== moved
        ps.ymd > '12345678' ;
```

This problem has been corrected in Oracle Rdb Release 7.0.6.2.

4.1.2 Excessive Pages Discarded when Using COMMIT TO JOURNAL OPTIMIZATION

Bug 1533127

When the COMMIT TO JOURNAL OPTIMIZATION was enabled and a READ ONLY transaction was active, Oracle Rdb would not reclaim space on data pages for deleted lines. For example, if an online backup operation was active, then for the duration of the backup operation, space would not be reclaimed. This could result in a high number of "pages discarded" as displayed on the "Record Statistics" screen of the RMU/SHOW STATISTICS Utility. It was also possible to see unneeded storage area extensions.

This problem has been corrected in Oracle Rdb Release 7.0.6.2. READ ONLY transactions no longer prevent Oracle Rdb from reclaiming deleted lines when the COMMIT TO JOURNAL feature is enabled.

4.1.3 Bugchecks at PIOGB$FETCH_FROM_GB + 488

Bug 714899

When the global buffer feature was enabled, it was possible to get bugchecks in PIOGB$FETCH_FROM_GB due to a deadlock between a page lock request and an Oracle Rdb internal buffer latch request.

This problem has been corrected in Oracle Rdb Release 7.0.6.2.

4.1.4 Query with CONCATENATE in BETWEEN Clause Returns Wrong Results

Bug 1663038

The following query uses the CONCATENATE function in the BETWEEN clause. It should return 3 rows, but it returns only 1 row.

```
SQL> sh tab ORDER;
Information for table ORDER

 Columns for table ORDER:
 Column Name                    Data Type        Domain
 -----------                    ---------        ------
 ORDER_NO                       CHAR(4)
  Not Null constraint ORDER_NO_NOT_NULL
 SHIP_DATE              CHAR(8)
  Not Null constraint ORDER_NOT_NULL
 SHIP_STAT              CHAR(1)
  Not Null constraint ORDER_NOT_NULL
...etc...

Table constraints for ORDER:
ORDER_NOT_NULL
 Not Null constraint
 Column constraint for ORDER.SHIP_DATE
 Evaluated on COMMIT
 Source:
        ORDER.SHIP_DATE NOT null
...etc...

SQL> select order_no from customer;
 ORDER_NO
 1ED0
 1j80
 1a78
3 rows selected
SQL> select order_no,ship_date,ship_stat from order;
 ORDER_NO   SHIP_DATE   SHIP_STAT
 1ED0       20010301    b
 1a78       20010228    a
 1j80       20010301    a
3 rows selected

set flags 'strategy,detail';
set flags 'max_stab';
select  a.order_no, a.ship_date, a.ship_stat
from    ORDER a, CUSTOMER b
where   a.order_no = b.order_no and
        ((a.SHIP_DATE || a.SHIP_STAT)
                BETWEEN '20010228a' '20010301d') ;
Tables:
  0 = ORDER
  1 = CUSTOMER
Cross block of 2 entries
  Cross block entry 1
    Conjunct:
    (0.SHIP_DATE > SUBSTRING ('20010228a' FROM 0 FOR 8)) OR
     ((0.SHIP_DATE = SUBSTRING ('20010228a' FROM 0 FOR 8)) AND
      (0.SHIP_STAT >= SUBSTRING ('20010228a' FROM 8)))
    Conjunct:
```

```
    ((0.SHIP_DATE < SUBSTRING ('20010301d' FROM 0 FOR 8)) AND
     NOT MISSING (0.SHIP_STAT)) OR
     ((0.SHIP_DATE = SUBSTRING ('20010301d' FROM 0 FOR 8)) AND
      (0.SHIP_STAT <= SUBSTRING ('20010301d' FROM 8)))
    Get     Retrieval by index of relation 0:ORDER
      Index name  ORDER_UM01 [0:0]
  Cross block entry 2
    Index only retrieval of relation 1:CUSTOMER
      Index name  CUSTOMER_UM01 [1:1]         Direct lookup
        Key: 0.ORDER_NO = 1.ORDER_NO
 A.ORDER_NO   A.SHIP_DATE   A.SHIP_STAT
 1a78         20010228              a
1 row selected
```

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

1. The table columns contain NOT NULL constraints.
2. The query contains a BETWEEN clause with CONCATENATE function on two columns.

As a workaround, the query works if the column constraint ORDER_NOT_NULL is removed from the columns of table ORDER.

This problem has been corrected in Oracle Rdb Release 7.0.6.2.

4.1.5 ORDER BY Query With GROUP BY on Two Joined Derived Tables Returns Wrong Results

Bug 1694233

The following query with GROUP BY and ORDER BY clauses on two joined derived tables returns the results in the wrong order.

```
set flags 'strategy,detail';

select
        cast (a.name as char(5)) as name,
        a.datum
  from (select name, datum,
               cast (count (*) as integer) as count_a
          from a
          group by name, datum) a
       join
       (select name, datum,
               cast (count (*) as integer) as count_b
          from b
          group by name, datum) b
       on     a.name = b.name
          and a.datum = b.datum
   group by a.name, b.name, a.datum, b.datum, count_a
   order by name desc, a.datum asc
  ;
Tables:
  0 = A
  1 = B
Reduce: 0.NAME, 0.DATUM, 1.NAME, 1.DATUM, CAST (<mapped field> AS INT)
Sort: 0.NAME(a), 0.DATUM(a), 1.NAME(a), 1.DATUM(a), CAST (<mapped field> AS INT)
      (a)
Cross block of 2 entries
  Cross block entry 1
    Merge of 1 entries
```

```
      Merge block entry 1
      Aggregate: COUNT (*)
      Sort: 0.NAME(a), 0.DATUM(a)
      Get Retrieval sequentially of relation 0:A
  Cross block entry 2
    Merge of 1 entries
      Merge block entry 1
      Aggregate: COUNT (*)
      Sort: 1.NAME(a), 1.DATUM(a)
      Conjunct: (0.NAME = 1.NAME) AND (0.DATUM = 1.DATUM)
      Get     Retrieval sequentially of relation 1:B
 A.NAME   A.DATUM
 AAAA      1-JAN-2000 00:00:00.00   <=== BBBB should be followed by AAAA
 BBBB      1-JAN-2000 00:00:00.00
2 rows selected
```

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

1. The main query contains a GROUP BY clause on the columns of the two joined derived tables with GROUP BY.
2. One of the columns from the derived tables is cast as the same data type.
3. The ORDER BY clause references the cast column but using descending order.

There is no known workaround for this problem.

This problem has been corrected in Oracle Rdb Release 7.0.6.2.

4.1.6 Left Outer Join Query With CONCATENATE Returns Wrong Results

Bug 1680135

The following left OJ query with CONCATENATE should return 1 row but instead returns 0 rows.

```
set flags 'strategy,detail';
SELECT ttt.entity_id,
       ttt.cpty_id,
       ttt.trade_count
FROM (SELECT tt.entity_id,
             tt.cpty_id,
             SUM (tt.trade_count) as trade_count
      FROM (SELECT df.entity_id,
                   df.cpty_id,
                   case
                     when df.deal_status = 'X' then 1 else 0
                   end as trade_count
            from deal_folder df) as tt
      GROUP BY tt.entity_id, tt.cpty_id) as ttt
    LEFT OUTER JOIN
     contact c ON (c.cpty_id = ttt.cpty_id)
 WHERE
    ttt.trade_count <> 0
    and ttt.entity_id || ttt.cpty_id  > ''  ! <== this is causing problem
    ;
Tables:
  0 = DEAL_FOLDER
  1 = CONTACT
Conjunct: (<mapped field> <> 0) AND ((0.ENTITY_ID || 0.CPTY_ID) > '') <=(1)
Cross block of 2 entries        (Left Outer Join)
  Cross block entry 1
```

```
    Conjunct: <mapped field> <> 0
    Merge of 1 entries
      Merge block entry 1
      Aggregate: SUM (CASE (WHEN (0.DEAL_STATUS = 'X') THEN 1
                  ELSE 0))
      Sort: 0.ENTITY_ID(a), 0.CPTY_ID(a)
      Merge of 1 entries
        Merge block entry 1
        Conjunct: (0.ENTITY_ID || 0.CPTY_ID) > ''
        Index only retrieval of relation 0:DEAL_FOLDER
          Index name  DEAL_FOLDER_MONITOR_IDX [0:0]
  Cross block entry 2
    Conjunct: (<mapped field> <> 0) AND ((0.ENTITY_ID || 0.CPTY_ID) > '') <=(2)
    Conjunct: 1.CPTY_ID = 0.CPTY_ID
    Index only retrieval of relation 1:CONTACT
      Index name  CONTACT_IDX [0:0]
0 rows selected
```

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

1. The main query is a left outer join between a derived table and a table.
2. The derived table contains a GROUP BY clause on the columns of another derived table with an aggregate function SUM as the output column.
3. The main query has a WHERE predicate containing the CONCATENATE function on two or more columns of the derived table.
4. The main query has another WHERE predicate which references the output column of the aggregate function from the derived table.

As a workaround, the query works if the table 1:CONTACT has some rows or the following CONCATENATE function is replaced by the following predicates:

```
    ttt.entity_id || ttt.cpty_id  > ''

    is replaced by

    ttt.entity_id > '' AND ttt.cpty_id > ''
```

This problem has been corrected in Oracle Rdb Release 7.0.6.2.

4.1.7 Query With UNION in German Collating Sequence Returns Wrong Results

Bug 1684612

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

```
select d.datum, d.id, d.team
from teamer d,
      (select s.datum,s.id, s.team
        from team_datum s
        union all
       select datum, id, team
        from team_datum
          ) as s
where
    d.datum=s.datum
      ;
```

```
Tables:
  0 = teamer
  1 = team_datum
  2 = team_datum
Conjunct: 0.datum = <mapped field>
Match
  Outer loop
    Sort: <mapped field>(a)
    Merge of 1 entries
      Merge block entry 1
      Merge of 2 entries
        Merge block entry 1
        Get Retrieval sequentially of relation 1:team_datum
        Merge block entry 2
        Get Retrieval sequentially of relation 2:team_datum
  Inner loop
    Temporary relation
    Sort: <mapped field>(a)
    Get Retrieval sequentially of relation 0:teamer
0 rows selected
```

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

1. The query is a simple join between a table and a derived table of subselects unioned together.
2. The join predicate uses CHAR data type.
3. The Optimizer uses a match strategy to join them, where a comparison of the join keys requires the process of encoding the CHAR data type into German collating sequence.

There is no known workaround for this problem.

This problem has been corrected in Oracle Rdb Release 7.0.6.2.

4.1.8 Query With OR Predicate on Aggregate Column Returns Wrong Results

Bugs 1708342 and 1721323

Query #1:

The following query with an OR predicate should return 1 row with T1.STATUS = 3 but returns an extra row with T1.STATUS = 5. This row does not satisfy the condition in the predicate "x.summe is null".

```
set flags 'max_stability';
set flags 'strategy,detail';
select
    t1.id,
    t1.status,
    t1.anzahl_stuecke,
    x.summe
  from table1 t1,
    (select sum(anzahl_stuecke) as summe
     from table2 t2
     where t1.id = t2.id ) x
  where
     t1.status = 3
     OR
     (t1.status = 5 and  x.summe is null) ;
Tables:
  0 = TABLE1
  1 = TABLE2
```

```
Cross block of 2 entries
  Cross block entry 1
    Conjunct: (0.STATUS = 3) OR (0.STATUS = 5)
    Get     Retrieval by index of relation 0:TABLE1
      Index name  XPKTABLE1 [0:0]
  Cross block entry 2
    Merge of 1 entries
      Merge block entry 1
      Aggregate: SM (1.ANZAHL_STUECKE)
      Get     Retrieval by index of relation 1:TABLE2
        Index name  XPKTABLE2 [1:1]
          Keys: 0.ID = 1.ID
       T1.ID     T1.STATUS   T1.ANZAHL_STUECKE                X.SUMME
           1             3                  10                   NULL
           2             5                  10                     10
2 rows selected
```

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

1. The main query joins a table and a derived table with a column of an aggregate function (e.g. SUM).
2. The WHERE clause contains an OR predicate, where one of the branches references the aggregated column.

As a workaround, the query works if the branches of the OR predicates are swapped, as in the following example.

```
select
    t1.id,
    t1.status,
    t1.anzahl_stuecke,
    x.summe
  from table1 t1,
    (select sum(anzahl_stuecke) as summe
     from table2 t2
     where t1.id = t2.id ) x
  where
     (t1.status = 5 and  x.summe is null)
     OR
     t1.status = 3 ;
Tables:
  0 = TABLE1
  1 = TABLE2
Cross block of 2 entries
  Cross block entry 1
    Get     Retrieval by index of relation 0:TABLE1
      Index name  XPKTABLE1 [0:0]
  Cross block entry 2
    Conjunct: ((0.STATUS = 5) AND MISSING (var) OR (0.STATUS = 3)
    Merge of 1 entries
      Merge block entry 1
      Aggregate: SM (1.ANZAHL_STUECKE)
      Get     Retrieval by index of relation 1:TABLE2
        Index name  XPKTABLE2 [1:1]
          Keys: 0.ID = 1.ID
       T1.ID     T1.STATUS   T1.ANZAHL_STUECKE                X.SUMME
           1             3                  10                   NULL
1 row selected
```

Query #2:

The following query with an OR predicate should return 0 rows.

```
set flags 'max_stability';
set flags 'strategy,detail';
select
    t1.id,
    t1.status,
    t1.anzahl_stuecke,
    x.summe
  from table1 t1,
    (select
        sum(anzahl_stuecke) as summe,
        'hello' as Artikel
     from table2 t2
     where t1.id = t2.id ) x
  where
    t1.id <> 5 and
     x.Artikel = 'hello should not be found' and
    ((t1.status =3) or
      (t1.status = 5 and (x.summe is NULL))
    );
Tables:
  0 = TABLE1
  1 = TABLE2
Cross block of 2 entries
  Cross block entry 1
    Get     Retrieval by index of relation 0:TABLE1
      Index name  XPKTABLE1 [0:0]
        Bool: 0.ID <> 5
  Cross block entry 2
    Conjunct: (0.STATUS = 3) OR ((0.STATUS = 5) AND MISSING (var)
    Merge of 1 entries
      Merge block entry 1
      Aggregate: SUM (1.ANZAHL_STUECKE)
      Get     Retrieval by index of relation 1:TABLE2
        Index name  XPKTABLE2 [1:1]
          Keys: 0.ID = 1.ID
          Bool: (1.ID <> 5) AND ('hello' = 'hello should not be found')
       T1.ID     T1.STATUS   T1.ANZAHL_STUECKE                X.SUMME
           1             3                  10                   NULL
           2             5                  10                   NULL
2 rows selected
```

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

1. The main query joins a table and a derived table with the column of an aggregate function (e.g. SUM) and a column of a constant string.
2. The WHERE clause contains an OR predicate, where one of the branches references the aggregate column.
3. The WHERE clause contains additional AND predicates where one of them references the column of a constant string.

There is no known workaround for this problem.

This problem has been corrected in Oracle Rdb Release 7.0.6.2.

4.1.9 Query With Equality Predicate Included in IN Clause Returns Wrong Results

Bug 1727181

The following query with an equality predicate included in the IN clause should find the row.

```
set flags 'strategy,detail';
select employee_id
  from employees e, departments d
  where
    e.employee_id = d.manager_id and
    d.department_code in ('ADMN', 'ENG', 'MKTG') and
    d.department_code = 'ENG'
    ;
Tables:
  0 = EMPLOYEES
  1 = DEPARTMENTS
Cross block of 2 entries
  Cross block entry 1
    Conjunct: (1.DEPARTMENT_CODE = 'ADMN') OR (1.DEPARTMENT_CODE = 'MKTG')
    Conjunct: 1.DEPARTMENT_CODE = 'ENG'
    Index only retrieval of relation 1:DEPARTMENTS
      Index name  DEPT_DEPTCODE_MGRID [1:1]
        Keys: 1.DEPARTMENT_CODE = 'ENG'
  Cross block entry 2
    Index only retrieval of relation 0:EMPLOYEES
      Index name  EMP_EMPID_STATUS_CODE [1:1]
        Keys: 0.EMPLOYEE_ID = 1.MANAGER_ID
0 rows selected
```

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

1. The query joins two tables using a join predicate.
2. The query has an equality predicate which is also included in the IN clause.

As a workaround, the query works if the equality predicate is moved to the front of the IN clause, as in the following example.

```
set flags 'strategy,detail';
select employee_id
  from employees e, departments d
  where
    e.employee_id = d.manager_id and
    d.department_code = 'ENG' and               <== move to front
    d.department_code in ('ADMN', 'ENG', 'MKTG')
    ;
Tables:
  0 = EMPLOYEES
  1 = DEPARTMENTS
Cross block of 2 entries
  Cross block entry 1
    Conjunct: 1.DEPARTMENT_CODE = 'ENG'
    Conjunct: (1.DEPARTMENT_CODE = 'ADMN') OR (1.DEPARTMENT_CODE = 'ENG') OR (
              1.DEPARTMENT_CODE = 'MKTG')
    Index only retrieval of relation 1:DEPARTMENTS
      Index name  DEPT_DEPTCODE_MGRID [1:1]
        Keys: 1.DEPARTMENT_CODE = 'ENG'
  Cross block entry 2
    Conjunct: 1.DEPARTMENT_CODE = 'ENG'
    Index only retrieval of relation 0:EMPLOYEES
      Index name  EMP_EMPID_STATUS_CODE [1:1]
        Keys: 0.EMPLOYEE_ID = 1.MANAGER_ID
 E.EMPLOYEE_ID
 00471
```

```
1 row selected
```

This problem has been corrected in Oracle Rdb Release 7.0.6.2.

4.1.10 Bugchecks at DIOCCHDBR$UNLATCH_GRCL With Exception of COSI−F−NONEXPR

In very rare cases of process failure when using the row cache feature, it was possible for an Oracle Rdb process or database recovery process (DBR) to fail with an exception of COSI−F−NONEXPR within DIOCCHDBR$UNLATCH_GRCL (typically at offset 0000034C). This problem was found during in−house high−load stress testing and was not customer reported.

This bugcheck was due to another process on the system being killed while waiting for a latch. The bugcheck was triggered when the original process attempted to wake the (now non−existant) process that had been waiting.

This problem has been corrected in Oracle Rdb Release 7.0.6.2. The unlatch code now correctly ignores the missing process during the wake request.

4.1.11 Match Strategy on Columns of Different Size, Using Collating Sequence, Returns Wrong Results

Bug 1684643

The following query using match strategy on columns of different size, using German collating sequence, should find the row.

```
select d.datum, d.abtlg, d.team, d.art
from teamergebnis_kumul d,
      (select m.datum,m.abtlg, m.art, m.team
       from std_team_datum m, prod_kumul_datum v
             where m.datum=v.datum and
            m.abtlg=v.abtlg and
            m.team=v.produkt AND
            m.team='11.3512'
        group by m.datum, m.abtlg, m.art, m.team) AS
        s (datum, abtlg, art, team)
where d.datum=s.datum and
      d.abtlg=s.abtlg and
      d.team=s.team and
      d.art=s.art and
      d.abtlg='465' and d.datum='20001031' and
      d.team='11.3512';
Tables:
  0 = TEAMERGEBNIS_KUMUL
  1 = STD_TEAM_DATUM
  2 = PROD_KUMUL_DATUM
Cross block of 2 entries
  Cross block entry 1
    Conjunct: 0.TEAM = '11.3512'
    Get     Retrieval by index of relation 0:TEAMERGEBNIS_KUMUL
      Index name  IDX_TEAMERGEBNIS_KUMUL_SORT [3:3]
        Keys: (0.TEAM = '11.3512') AND (0.DATUM = '20001031') AND (0.ABTLG =
              '465')
  Cross block entry 2
    Conjunct: 0.ABTLG = 1.ABTLG
    Conjunct: 0.TEAM = 1.TEAM
```

```
    Conjunct: 0.ART = 1.ART
    Merge of 1 entries
      Merge block entry 1
      Reduce: 1.TEAM, 1.ABTLG, 1.DATUM, 1.ART
      Sort: 1.TEAM(a), 1.ABTLG(a), 1.DATUM(a), 1.ART(a)
      Conjunct: (1.DATUM = 2.DATUM) AND (1.ABTLG = 2.ABTLG) AND (1.TEAM =
                2.PRODUKT)
      Match
        Outer loop
          Sort: 1.TEAM(a), 1.ABTLG(a), 1.DATUM(a)
          Conjunct: 1.TEAM = '11.3512'
          Get     Retrieval by index of relation 1:STD_TEAM_DATUM
            Index name  IDX_STD_TEAM_DATUM_SORT [2:2]
              Keys: (0.DATUM = 1.DATUM) AND (1.ABTLG = '465')
        Inner loop
          Temporary relation
          Sort: 2.PRODUKT(a), 2.ABTLG(a), 2.DATUM(a)
          Conjunct: 2.PRODUKT = '11.3512'
          Get     Retrieval by index of relation 2:PROD_KUMUL_DATUM
            Index name  IDX_PROD_KUMUL_DATUM_SORT [2:2]
              Keys: (2.DATUM = 0.DATUM) AND (2.ABTLG = '465')
0 rows selected
```

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

1. The main query is a simple join between a table and a derived table of subselect subquery, joining two tables using 3 equality predicates.
2. The join predicate uses columns of CHAR data type but different column size.
3. The optimizer uses a match strategy to join them, where a comparison of the join keys requires the process of encoding the CHAR data type into German collating sequence.

As a workaround, the query works if the match strategy is changed to use cross by using an outline.

This problem has been corrected in Oracle Rdb Release 7.0.6.2.

4.1.12 Network Link Failure Does Not Allow DISCONNECT to Clean Up Transactions

Bug 856747

In earlier versions of Oracle Rdb, if a program attached to a database on a remote node and it lost the connection before the COMMIT statement was issued, there was nothing you could do except exit the program and start again.

It is now possible to DISCONNECT the database and reconnect without restarting the program.

The following example shows a dynamic SQL session to a remote database which loses its connection to the remote server:

```
SQL> attach 'filename ataxp1::dkb200:[scott]personnel';
SQL> select * from rdb$database;

-- at this point connection to remote server is lost:

Error -1 returned from open_cursor
Error message:
%RDB-F-IO_ERROR, input or output error
-SYSTEM-F-LINKABORT, network partner aborted logical link
SQL> rollback;
```

```
Error message:
%RDB-F-IO_ERROR, input or output error
-SYSTEM-F-LINKABORT, network partner aborted logical link
SQL> disconnect current;
Error message:
%RDB-F-IO_ERROR, input or output error
-SYSTEM-F-LINKABORT, network partner aborted logical link
```

The next example shows how to recover now that the disconnect is possible:

```
SQL> attach 'filename ataxp1::dkb200:[scott]personnel';
SQL> select * from rdb$database;

-- at this point connection to remote server is lost:
Error -1 returned from open_cursor
Error message:
%RDB-F-IO_ERROR, input or output error
-SYSTEM-F-LINKABORT, network partner aborted logical link
SQL> disconnect current;
SQL> attach 'filename ataxp1::dkb200:[scott]personnel';
SQL>
-- continue working
```

This problem has been corrected in Oracle Rdb Release 7.0.6.2.

4.1.13 Failure to Extend a Storage Area May Leave the LEOF of the .RDA File Pointing Beyond the PEOF

Bug 1316670

The logical EOF (LEOF) of the storage area file could be pointing beyond the physical EOF (PEOF) of the file if an attempt to extend the storage area fails. This happened since the LEOF was set to the new value even though the extend of the file failed.

This problem has been corrected in Oracle Rdb Release 7.0.6.2. Now the LEOF is set to the new value only if the extend operation succeeds, for example, the PEOF changes.

4.1.14 Left Outer Join Query With CAST Function on USING Column Bugchecks

Bug 1802653

The following left outer join query with CAST function on USING column bugchecks.

```
select count(*) from
( select p.paketwert from
  ( select
        cast(packet as integer)   ! <=== CAST causing bugcheck
    from
    serien inner join sujet using (sujet)
  ) as p (paketwert)
) as astpreis (paketwert)
left outer join
( select t.paketwert from
  ( select
        packet
    from
```

```
    serien inner join sujet using (sujet)
  ) as t (paketwert)
) as opt(paketwert)
USING (paketwert) ;
```

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

1. The main query is a left outer join of 2 nested derived tables.
2. The CAST function is placed on the column of USING clause.

There is no known workaround for this problem.

This problem has been corrected in Oracle Rdb Release 7.0.6.2.

4.1.15 Query Using Constant Values in OR Predicates Returns Wrong Results

Bug 1769447

The following query using constant values in OR predicates should return 3 rows.

```
set flags 'strategy,detail';

SELECT  col1 FROM
 (SELECT
     t2.col1 as col1,
     t2.col2 as col2,
     t2.col3 as col3
     from table1 t1, table2 t2
     where t1.col1_id = t2.col1_id
     ) as
   vt (col1, col2, col3)
WHERE
        vt.col3 > 0  AND
        vt.col2 >= 0       AND
        ( vt.col1 <= 3 OR 'hostvar' = 'foo' );
Tables:
  0 = TABLE1
  1 = TABLE2
Merge of 1 entries
  Merge block entry 1
  Conjunct: 0.col1_id = 1.col1_id
  Match
    Outer loop      (zig-zag)
      Index only retrieval of relation 0:TABLE1
        Index name  TABLE1_NDX [0:0]
    Inner loop      (zig-zag)
      Conjunct: (1.col3 > 0) AND (1.col2 >= 0)
      Get     Retrieval by index of relation 1:TABLE2
        Index name  TABLE2_NDX [0:0]
          Bool: <error: common keyonly boolean no predicates>
        COL1
           1
           2
           3
           4
           5
           6
6 rows selected
```

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

1. The query selects from a derived table of a subselect joining 2 tables.
2. The WHERE clause contains 2 AND predicates and 1 OR predicate.
3. The OR predicate contains a branch of constant predicates, such as "1 = 2".

As a workaround, the query works if the constant condition "'hostvar' = 'foo'" is omitted, as in the following example.

```
set flags 'strategy,detail';

SELECT  col1 from
 (SELECT
     t2.col1 as col1,
     t2.col2 as col2,
     t2.col3 as col3
     from table1 t1, table2 t2
     where t1.col1_id = t2.col1_id
     ) as
   vt (col1, col2, col3)
WHERE
        vt.col3 > 0  AND
        vt.col2 >= 0       AND
        ( vt.col1 <= 3
!           OR 'hostvar' = 'foo'              <=== commented out
            );
Tables:
  0 = TABLE1
  1 = TABLE2
Merge of 1 entries
  Merge block entry 1
  Conjunct: 0.col1_id = 1.col1_id
  Match
    Outer loop      (zig-zag)
      Index only retrieval of relation 0:TABLE1
        Index name  TABLE1_NDX [0:0]
    Inner loop      (zig-zag)
      Conjunct: (1.col3 > 0) AND (1.col2 >= 0) AND (1.col1 <= 3)
      Get     Retrieval by index of relation 1:TABLE2
        Index name  TABLE2_NDX [0:0]
          Bool: 1.col1 <= 3
        COL1
           1
           2
           3
3 rows selected
```

This problem has been corrected in Oracle Rdb Release 7.0.6.2.

4.1.16 Manual Open Causes Utility Access State to Persist Until Close

Bug 1587509

If a database was implicitly opened via some database utility access such as a backup and, while the utility was attached to the database, a manual open was issued, then the database would retain the "utility access only" state.

For example, if a database was opened by an RMU/BACKUP/ONLINE command, while the backup was

executing an RMU/SHOW SYSTEM would display the following state for the database:

```
* database is available for utility access only
```

If an RMU/OPEN was then issued for the database while the backup was executing then the above message would be displayed even after the backup process completed. Having the database in this state would prevent the automatic startup of database servers such as the AIJ Log Server (ALS) or Row Cache Server (RCS). The processes could still be manually started by the RMU/SERVER START command.

This problem has been corrected in Oracle Rdb Release 7.0.6.2. Now when an RMU/OPEN command is issued the "utility access only" state will be cleared.

4.1.17 LogMiner Compresses Pre−Delete Record Content

Previously, when the Oracle Rdb LogMiner(TM) feature was enabled, the pre−delete record contents were not compressed prior to being journaled. Because of this, it was possible for AIJ files to grow excessively if many large records were being deleted.

This problem has been corrected in Oracle Rdb Release 7.0.6.2. When the Oracle Rdb LogMiner feature is enabled, pre−delete record contents are now correctly compressed. Because of the difference in pre−delete record contents in an AIJ file, it is important that AIJ files created with prior versions of Oracle Rdb be processed with the matching version of the Oracle Rdb LogMiner (RMU /UNLOAD /AFTER_JOURNAL command).

When using the Oracle Rdb LogMiner feature, existing AIJ files should be backed up and processed prior to upgrading to this release of Oracle Rdb.

Failure to use the correct version of the Oracle Rdb LogMiner to process an AIJ file typically results in RMU−W−RECVERDIF warnings when pre−delete record contents are being processed.

LogMiner AIJ files not compatible

When the Oracle Rdb LogMiner(TM) feature is being used, AIJ files from this version of Oracle Rdb are not compatible with the Oracle Rdb LogMiner feature from prior versions of Oracle Rdb. Only the Oracle Rdb LogMiner feature is affected; AIJ recovery is not affected. If the Oracle Rdb LogMiner feature is not enabled for a database, there is no difference in the format or content of an AIJ file.

4.1.18 Excessive Disk I/O for DROP TABLE and TRUNCATE TABLE

Bug 989292

In prior releases of Oracle Rdb, the DROP TABLE and TRUNCATE TABLE statements performed excessive disk I/O when the table contained LIST OF BYTE VARYING columns. When this data type is present, these operations must read the table to locate the LIST data. In prior releases, a DELETE operation was also performed on the table. While this achieved the delete of the LIST data, it also caused constraints (and possibly triggers) to be executed as well as updating indices as each row was deleted.

This problem was corrected in Oracle Rdb Release 7.0.4 (but the note was inadvertently omitted from the 7.0.4 Release Notes). The DROP TABLE and TRUNCATE TABLE statements no longer cause constraints and triggers to be executed for the table, and indices are no longer updated when processing the LIST OF

BYTE VARYING columns. The result is that I/O required for DROP TABLE and TRUNCATE TABLE is significantly reduced, especially for tables stored in UNIFORM format storage areas.

4.1.19 Query Joining Derived Tables of Union Legs With Empty Tables Returns Wrong Results

Bug 1818374

The following query, joining two derived tables containing union legs with empty tables, returns wrong results of 0 rows, instead of 1 row.

```
set flags 'strategy,detail';
select c1
  from (select v1.c1 from
        t_02,
        (select * from t_01
            union all
         select * from t_02
         ) v1
         inner join
        (select * from tt_01
            union all
         select * from tt_02
         ) as v2
        on (v1.c1 = v2.c1 and v1.c2 = v2.c2)) as tmp
  where tmp.c1 = 110759;
Tables:
  0 = T_02
  1 = T_01
  2 = T_02
  3 = TT_01
  4 = TT_02
Merge of 1 entries
  Merge block entry 1
  Cross block of 3 entries
    Cross block entry 1
      Index only retrieval of relation 0:T_02
        Index name  T_02_NDX [0:0]
    Cross block entry 2
      Merge of 1 entries
        Merge block entry 1
        Merge of 2 entries
          Merge block entry 1
          Conjunct: 1.C1 = 110759
          Index only retrieval of relation 1:T_01
            Index name  T_01_NDX [1:1]
              Keys: <mapped field> = 110759
          Merge block entry 2
          Leaf#01 FFirst 2:T_02 Card=1
            Bool: 2.C1 = 110759
            BgrNdx1 T_02_NDX [1:1] Fan=17
              Keys: <mapped field> = 110759
    Cross block entry 3
      Conjunct: 1.C1 = 110759
      Merge of 1 entries
        Merge block entry 1
        Merge of 2 entries
          Merge block entry 1
          Conjunct: (<mapped field> = 3.C1) AND (<mapped field> = 3.C2)
          Index only retrieval of relation 3:TT_01
            Index name  TT_01_NDX [2:2]
              Keys: (<mapped field> = <mapped field>) AND (<mapped field> =
```

```
                    <mapped field>)
          Merge block entry 2
          Conjunct: (<mapped field> = 4.C1) AND (<mapped field> = 4.C2)
          Index only retrieval of relation 4:TT_02
            Index name  TT_02_NDX [2:2]
              Keys: (<mapped field> = <mapped field>) AND (<mapped field> =
                    <mapped field>)
0 rows selected

where the tables are defined as :

! table t_01 is empty
create table t_01 (C1   INTEGER);
create index t_01_ndx  on t_01 (C1) ;

! table t_02 has 1 row
create table t_02 (C1  INTEGER, C2   TINYINT);
create index t_02_ndx  on t_02 (C1) ;

insert into t_02  values (110759,9);

! table tt_01 is empty
create table tt_01 (C1  INTEGER, C2   TINYINT);
create index tt_01_ndx on tt_01 (C1,  C2);

! table tt_02 has 2 rows
create table tt_02 (C1  INTEGER, C2   TINYINT);
create index tt_02_ndx on tt_02 (C1,  C2);

insert into tt_02 values (110759,4);
insert into tt_02 values (110759,9);
```

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

1. The main query selects the column of a derived table with an equality predicate.
2. The main derived table joins a non−empty table (t_02) and an inner join.
3. The inner join involves a derived table of union between an empty table (t_01) and a non−empty table (t_02), and another derived table of union between an empty table (tt_01) and a non−empty table (tt_02).

As a workaround, the query works if the empty tables are loaded with some data as in the following example.

```
insert into t_01 values (110759);

select c1
  from (select v1.c1 from
        t_02,
        (select * from t_01
            union all
         select * from t_02
         ) v1
         inner join
        (select * from tt_01
            union all
         select * from tt_02
         ) as v2
        on (v1.c1 = v2.c1 and v1.c2 = v2.c2)) as tmp
  where tmp.c1 = 110759;
          C1
      110759
1 row selected
```

This problem has been corrected in Oracle Rdb Release 7.0.6.2.

4.1.20 Left Outer Join Query With OR Predicate Returns Wrong Results

Bug 1837522

The following left outer join query with an OR predicate, having an equality predicate of a column and a constant value on the left side, and an equality predicate of a column and a subquery on the right side, returns wrong results. It should find 3 rows, but it only finds 2 rows.

```
set flags 'strategy,detail';
select job_code, job_start, c1.employee_id, c2.employee_id
  from
  job_history as c1
  left outer join
  employees as c2 on (c1.employee_id = c2.employee_id)
  where
      c1.job_code = 'JNTR' or
      c1.job_start =
      (select max(job_start) from job_history as c3)
      ;
Tables:
  0 = JOB_HISTORY
  1 = EMPLOYEES
  2 = JOB_HISTORY
Cross block of 2 entries
  Cross block entry 1
    Aggregate: 0:MAX (2.JOB_START)
    Get     Retrieval by index of relation 2:JOB_HISTORY
      Index name  JH_EMPLOYEE_ID [0:0]
  Cross block entry 2
    Conjunct: 0.JOB_START = <agg0>
    Conjunct: 0.JOB_START = <agg0>
    Match    (Left Outer Join)
      Outer loop
        Conjunct: (0.JOB_CODE = 'JNTR') OR (0.JOB_START = <agg0>)
        Get     Retrieval by index of relation 0:JOB_HISTORY
          Index name  JH_EMPLOYEE_ID [0:0]
      Inner loop      (zig-zag)
        Index only retrieval of relation 1:EMPLOYEES
          Index name  EMP_EMPLOYEE_ID [0:0]
 C1.JOB_CODE   C1.JOB_START   C1.EMPLOYEE_ID   C2.EMPLOYEE_ID
 PRSD           3-Jan-1983    00225            00225
 DMGR           3-Jan-1983    00241            00241
2 rows selected
```

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

1. The main query is a left outer join between 2 tables with an ON clause.
2. The WHERE clause contains an OR predicate, with the left side branch being a simple equality predicate on a column, and the right branch using a sub-query in the equality predicate.

As a workaround, the query works if the left and right side of the OR predicate is swapped. For example:

```
select job_code, job_start, c1.employee_id, c2.employee_id
  from
  job_history as c1
  left outer join
```

```
  employees as c2
  on (c1.employee_id = c2.employee_id)
      where
      c1.job_start =
      (select max(job_start) from job_history as c3)
      or
      c1.job_code = 'JNTR'
      ;
 C1.JOB_CODE   C1.JOB_START   C1.EMPLOYEE_ID   C2.EMPLOYEE_ID
 JNTR           2-Jan-1977    00223            00223
 PRSD           3-Jan-1983    00225            00225
 DMGR           3-Jan-1983    00241            00241
3 rows selected
```

This problem has been corrected in Oracle Rdb Release 7.0.6.2.

4.1.21 Query Using Match Strategy With DISTINCT Function Returns Wrong Results

Bugs 1891938 and 1894192

A query using the match strategy with the Distinct Function returns the wrong results, as in the following example.

```
set flags 'strategy,detail';
select count(*) from
( select distinct
                  t1.ACCOUNT_ID,
                  t1.SECURITY_ID
         from     T1 t1,
                  T2 t2
         where    t1.SECURITY_ID = t2.SECURITY_ID
) as t ;
Tables:
  0 = T1
  1 = T2
Merge of 1 entries
  Merge block entry 1
  Reduce: 0.SECURITY_ID, 0.ACCOUNT_ID
  Sort: 0.SECURITY_ID(a), 0.ACCOUNT_ID(a)
  Conjunct: 0.SECURITY_ID = 1.SECURITY_ID
  Match
    Outer loop
      Sort: 1.SECURITY_ID(a)
      Get Retrieval sequentially of relation 1:T2
    Inner loop (zig-zag)
      Index only retrieval of relation 0:T1
        Index name  T1_NDX1 [0:0]
 ACCOUNT_ID   SECURITY_ID
 A1           DE0005557508
1 row selected

where the tables are defined as :

create table T1 (
    ACCOUNT_ID       CHAR (2),
    SECURITY_ID      CHAR (12) );
create index T1_NDX    on T1 (ACCOUNT_ID, SECURITY_ID);

create table T2 (SECURITY_ID      CHAR (12) );

with the following contents:
```

```
select SECURITY_ID from T2;

SECURITY_ID
 DE0005128003
 DE0005557508
2 rows selected

select ACCOUNT_ID,SECURITY_ID from T1;
ACCOUNT_ID   SECURITY_ID
 A1          DE0005557508
 PP          DE0005128003
2 rows selected
```

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

1. The main query selects from a derived table.
2. The derived table is the output of a distinct query from T1 and T2 with a join column predicate.
3. The join column of table T1 is the second segment in index T1_NDX which is ordered by the first segment ACCOUNT_ID.
4. The order of the join column of table T2 is ascending and different from that of T2.

As a workaround, the query works if the query outline is used to apply cross strategy instead of match, as in the following example.

```
select * from
( select
         distinct
                  t1.ACCOUNT_ID,
                  t1.SECURITY_ID
         from     T1 t1,
                  T2 t2
         where    t1.SECURITY_ID = t2.SECURITY_ID
) as t ;
~S: Outline "QO_325EFDCDDEBFFFA8_00000000" used
Tables:
  0 = T1
  1 = T2
Merge of 1 entries
  Merge block entry 1
  Reduce: 0.ACCOUNT_ID, 0.SECURITY_ID
  Sort: 0.ACCOUNT_ID(a), 0.SECURITY_ID(a)
  Cross block of 2 entries
    Cross block entry 1
      Get Retrieval sequentially of relation 1:T2
    Cross block entry 2
      Conjunct: 0.SECURITY_ID = 1.SECURITY_ID
      Index only retrieval of relation 0:T1
        Index name  T1_NDX [0:0]
-- Rdb Generated Outline : 31-JUL-2001 11:23
create outline QO_325EFDCDDEBFFFA8_00000000
id '325EFDCDDEBFFFA85200828890C4E5BA'
mode 0
as (
  query (
-- For loop
    subquery (
      subquery (
        T2 1    access path sequential
          join by cross to              -- <=== change from match to cross
        T1 0    access path index       T1_NDX
        )
      )
```

```
    )
  )
compliance optional     ;
 ACCOUNT_ID   SECURITY_ID
 A1           DE0005557508
 PP           DE0005128003
2 rows selected
```

This problem has been corrected in Oracle Rdb Release 7.0.6.2.

4.1.22 GROUP BY Query With SUM Aggregate Returns Wrong Results

Bug 1844260

The following GROUP BY query with SUM aggregate returns wrong results (the first row of column ESTADO should be 'A' instead of 'V').

```
set flags 'strategy,detail';
select estado, sum(total_dep) from bug_view group by estado;
Tables:
  0 = T1
  1 = T2
Aggregate: 0:SUM (CASE (WHEN (0.ID_PRODUCTO = 20) THEN 20 ELSE 15))
Sort: CASE (WHEN (1.FEC_EXPIRACION > 20001231) THEN 'A' WHEN (((0.ID_PRODUCTO =
      15) OR (0.ID_PRODUCTO = 20)) AND (1.FEC_EXPIRACION <= 20001231)) THEN 'V'
       ELSE NULL)(a)
Conjunct: 0.ID_PRODUCTO = 1.ID_PRODUCTO
Match
  Outer loop      (zig-zag)
    Index only retrieval of relation 0:T1
      Index name  T1_NDX [0:0]
  Inner loop      (zig-zag)
    Get     Retrieval by index of relation 1:T2
      Index name  T2_NDX [0:0]
 ESTADO
 V                          15          <=== ESTADO should be 'A'
 V                          15
2 rows selected

where the view is defined as :

create view bug_view ( id_producto, total_dep, estado ) as
select
         a.id_producto,
         case
             when a.id_producto = 20  then 20
             else 15
             end as total_dep,
         case
             when b.fec_expiracion > 20001231 then  'A'
             when (a.id_producto = 15
                   OR a.id_producto = 20
                   ) and
                  b.fec_expiracion <= 20001231
             then  'V'
             end  as estado
     from opas_saldos_err a, ope_pasiva_err b
     where
              a.id_producto = b.id_producto ;

with the following content in the tables:
```

```
select * From t1;
 ID_PRODUCTO
           8
1 row selected

select * From t2;
 ID_PRODUCTO   FEC_EXPIRACION
           8         20000801
           8         20010628
2 rows selected
```

As a workaround, the query works if the predicate "OR a.id_producto = 20" is commented out from the view, as in the following example.

```
create view bug_view_good ( id_producto, total_dep, estado ) as
select
         a.id_producto,
         case
             when a.id_producto = 20  then 20
             else 15
             end as total_dep,
         case
             when b.fec_expiracion > 20001231 then  'A'
             when (a.id_producto = 15
!                 OR a.id_producto = 20
                  ) and
                 b.fec_expiracion <= 20001231
             then  'V'
             end  as estado
     from t1 a, t2 b
     where
              a.id_producto = b.id_producto ;

select estado, sum(total_dep) from bug_view_good group by estado;
Tables:
  0 = T1
  1 = T2
Aggregate: 0:SUM (CASE (WHEN (0.ID_PRODUCTO = 20) THEN 20 ELSE 15))
Sort: CASE (WHEN (1.FEC_EXPIRACION > 20001231) THEN 'A' WHEN ((0.ID_PRODUCTO =
      15) AND (1.FEC_EXPIRACION <= 20001231)) THEN 'V' ELSE NULL)(a)
Conjunct: 0.ID_PRODUCTO = 1.ID_PRODUCTO
Match
  Outer loop      (zig-zag)
    Index only retrieval of relation 0:T1
      Index name  T1_NDX [0:0]
  Inner loop      (zig-zag)
    Get     Retrieval by index of relation 1:T2
      Index name  T2_NDX [0:0]
 ESTADO
 A                          15
 V                          15
2 rows selected
```

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

1. The main query contains a GROUP BY clause and SUM aggregate function.
2. The SUM aggregate function is defined in the view as a CASE expression.
3. The column in the GROUP BY clause is defined in the view as a CASE expression which contains the same predicate from the CASE expression of the SUM aggregate.

This problem has been corrected in Oracle Rdb Release 7.0.6.2.

4.1.23 ARBs Exhausted

It was possible for a database to run out of AIJ Request Blocks (ARBs) if many processes were abnormally terminated. If a process had an ARB allocated at the time it was terminated, the Database Recovery Process (DBR) would fail to free the ARB allocated to the process. This problem was introduced in Oracle Rdb Release 7.0.1.2.

Symptoms of this problem include:

- Processes looping. RMU/SHOW STATISTICS would show processes stalling waiting for the AIJ lock or writing the same AIJ block over and over.
- More AIJ activity due to processes flushing the ARBs more often in attempts to make ARBs available.
- The "AIJ Journal Information" screen displayed by RMU/SHOW STATISTICS would show available ARB count "(ARB.Avail:)" to be few or none.

To avoid the problem, avoid terminating processes via the DCL STOP /IDENTIFICATION command. When the problem occurs, the database must be closed and re−opened on each node where the problem is being seen to reset the free ARB lists.

This problem has been corrected in Oracle Rdb Release 7.0.6.2.

4.1.24 CLEAN BUFFER COUNT Parameter Not Obeyed

When the Asynchronous Batch Write feature is being used, Oracle Rdb is supposed to inspect the tail of the least recently used (LRU) buffer queue to determine if there are any modified buffers at the end of the queue. The CLEAN BUFFER COUNT parameter specifies how many buffers are to be inspected. If any are found, then those buffers are supposed to be written to disk. However, when unmarking buffers, Oracle Rdb would unmark buffers at the end of the modified queue instead of the LRU queue. That could cause buffers that were just modified to be immediately written, even if they were the most recently accessed buffers. This could cause the buffer to have to be modified again and thus written again.

This problem has been corrected in Oracle Rdb Release 7.0.6.2. Instead of writing the buffers at the tail of the modified queue, Oracle Rdb now writes the modified buffers at the end of the LRU queue.

4.1.25 DETECTED ASYNC PREFETCH THRESHOLD Not Obeyed

The detected async prefetch (DAPF) feature is supposed to initiate async prefetch (APF) requests if it detects consecutive pages being fetched from a storage area. The THRESHOLD parameter declares how many consecutive buffers read in a sequence will trigger an APF request. However, Oracle Rdb would not actually initiate APF requests until the THRESHOLD count plus half the DEPTH number of buffers were sequentially read.

This problem has been corrected in Oracle Rdb Release 7.0.6.2. DAPF will now be triggered when THRESHOLD number of consecutive buffers are read in a sequence.

4.1.26 Page Locks Not Demoted at End of Transaction When FAST COMMIT Enabled

When using the FAST COMMIT feature, at the end of a transaction, page locks were not being demoted. Page locks are always demoted at the end of a transaction when the FAST COMMIT feature is not enabled. In some applications, demoting page locks at the end of a transaction can significantly reduce the incidence of

deadlocks involving page locks.

This situation has been improved in Oracle Rdb Release 7.0.6.2. When the FAST COMMIT feature is enabled, at the end of a transaction any buffer that does not contain a modified page will have its page locks demoted.

4.1.27 Incorrect Record Written to AIJ for Ranked Indexes

Under some rare circumstances, an update to a ranked index entry may cause an incorrectly formatted record to be written to the after image journal, which may cause problems on subsequent restoration of that index.

This problem may occur if all the following are true:

1. An insert is made into a ranked index that causes a entry to change from unique to a duplicate.
2. Insufficient room is left on the index node for the insertion causing the index node to split.
3. Another process requests access to the same page cluster that contains the node the first process is updating after the time at which the first process has started making the modifications but prior to the completion of the index node split.
4. After image journaling is enabled.

This index problem will probably manifest itself as a bugcheck on recovery of this index from the corrupted AIJ file.

Bugchecks will have reference to one or more ranked index routines, 'PSII2*', for example:

```
PSII2SCANGETNEXTBBCDUPLICATE + 00000093
```

It is not possible to give a complete bugcheck footprint as it depends on the next action taken on that index during or after restoration.

The following is an example of a index dump of the node affected:

```
            46 14 07 0023  0303  7 bytes stored, 20 byte prefix
00571111111800000800043004843 00  pfx '..............W.'
                09800000          pfx '....'
  500F088000BD0F           0308   key     '......W'
           0536990617 75  030F   overflow pointer 83:432389:22
                    0002  0315   entry cardinality 2.
                    0000  0317   leaf cardinality 0.
                 0600 0C  0317   reference pointer 0:5:-1
                    0715  031A   1813 byte bitmap containing 3614921436 records
      0000 00000005 FFF8  031C    duplicate record 0:5:-8
      0000 00000005 FFF9  031C    duplicate record 0:5:-7
      0000 00000005 FFFA  031C    duplicate record 0:5:-6
      0000 00000005 FFFB  031C    duplicate record 0:5:-5
      0000 00000005 FFFC  031C    duplicate record 0:5:-4
                 ...
```

Note the incorrect 'bitmap containing' count and invalid reference pointer and dbkey values.

This problem has been corrected in Oracle Rdb Release 7.0.6.2.

4.1.28 ROLLBACK Hangs Under DECdtm When Called From an ACMS CANCEL Procedure

Under certain situations, the CANCEL procedure in an ACMS application would cause the ACMS server process to hang in the Rdb dispatch layer. This problem can only occur under the following circumstances:

1. The ACMS application is using 2 phase commit under DECdtm either explicitly (i.e. with a SYS$START_TRAN call) or implicitly (by attaching to multiple Rdb databases).
2. The CANCEL procedure contains a SYS$ABORT_TRAN call or ROLLBACK statement.
3. The ACMS server process has a outstanding pending request which is blocked (e.g. waiting for rows locked by another user).

If all three of these occured, the ACMS server process would hang in the CANCEL procedure even after the condition that caused the original blocking cleared.

This problem has been corrected in Oracle Rdb Release 7.0.6.2.

4.2 SQL Errors Fixed

4.2.1 Supplied CHR Function Returns Incorrect Value

The CHR function supplied as part of the SQL$FUNCTIONS script incorrectly returns NULL (when the dialect is set to ORACLE LEVEL1), or a zero length string (for other dialects). It should return a CHAR(1) string containing the ASCII character NUL. The external function definition was causing the zero character to be interpreted as a C null termination and so Rdb thought the result was an empty string.

The following SQL commands can be executed to replace the definition of the CHR external function so that it will return the correct value.

```
DROP FUNCTION CHR CASCADE;
CREATE FUNCTION CHR ( in INTEGER by reference )
    RETURNS RDB$ORACLE_SQLFUNC_CHAR_DOM;
    EXTERNAL NAME SQL$FNC_CHR
    LOCATION 'SQL$FUNCTIONS'
    WITH ALL LOGICAL_NAME TRANSLATION
    LANGUAGE GENERAL
    GENERAL PARAMETER STYLE
    NOT VARIANT
    COMMENT IS 'Returns the character having the binary equivalent to N. ';
GRANT EXECUTE ON FUNCTION CHR TO PUBLIC;
```

This problem has been corrected in Oracle Rdb Release 7.0.6.2.

4.2.2 IMPORT DATABASE Did Not Substitute New Collating Sequence

Bug 1695289

In prior releases of Oracle Rdb, IMPORT DATABASE would not handle changes to the collating sequence as expected.

- Specifying the existing database collating sequence on the IMPORT DATABASE command line would result in a reported error.

  ```
  SQL> import database
  cont>    from COLLATION_70.RBR
  cont>    filename 'COLLATE_OLD'
  cont>    collating sequence GERMAN GERMAN;
     .
     .
     .
  %SQL-F-NOCOLRES, Unable to import collating sequence GERMAN
  %RDB-E-NO_META_UPDATE, metadata update failed
  -RDMS-F-COLEXTS, there is another collating sequence named GERMAN in this
  database
     .
     .
     .
  ```

  This occurred because SQL would try to retain the old collating sequence in the database for reference purposes. If there was no change in the collating sequence name, SQL should have discarded the old

definition.

- Specifying a replacement database collating sequence on the IMPORT DATABASE command line would not cause it to replace the old collating sequence for domains in the new database.

These problems have been corrected in Oracle Rdb Release 7.0.6.2. IMPORT DATABASE now substitutes the new database collating sequence for the old in all imported domain definitions and IMPORT no longer generates an error if the COLLATING SEQUENCE clause re−specifies the existing database collating sequence.

Note

The original database collating sequence is imported for use in future domain definitions. If that collating sequence is no longer required, it can be dropped using the DROP COLLATING SEQUENCE statement.

The following example shows that the collating sequence is now changed by IMPORT DATABASE.

```
SQL> create database
cont>     filename COLLATE_GERMAN
cont>     collating sequence GERMAN GERMAN;
SQL> create domain NAMES_DOM
cont>     char(15);
SQL>
SQL> show domain names_dom;
NAMES_DOM                       CHAR(15)
 Collating sequence: GERMAN
SQL>
SQL> export database
cont>    alias rdb$dbhandle
cont>    into collation_70;
SQL> disconnect all;
SQL>
SQL> import database
cont>     from COLLATION_70.RBR
cont>     filename 'COLLATE_FRENCH'
cont>     collating sequence FRENCH FRENCH;
   .
   .
   .
Database collating sequence was GERMAN, now is FRENCH
   .
   .
   .
IMPORTing table EMPLOYEES
SQL>
SQL> show domain NAMES_DOM
NAMES_DOM                       CHAR(15)
 Collating sequence: FRENCH
SQL>
```

4.2.3 ATOMIC Block Not Rolling Back Changes on Error

In prior versions of Oracle Rdb 7.0, it was possible for updates made within an ATOMIC compound statement to be saved by COMMIT even when an exception was raised within that atomic block.

This could happen if INSERT, UPDATE or DELETE was performed by nested stored procedures which also included calls to nested stored procedures. The nested calls to procedures were losing the current ATOMIC state of the original compound statement.

The following simple example shows the code structure in which this problem could occur.

```
SQL> create table t1 (f1 int);
SQL>
SQL> create module m1
cont>     language sql
cont>
cont>     procedure p1;
cont>         begin not atomic
cont>         insert into t1 values (1);
cont>         end;
cont>
cont>     procedure p2;
cont>         begin not atomic
cont>         call p1();
cont>         end;
cont>
cont> end module;
SQL>
SQL> -- show that there are now rows
SQL> select * from t1;
0 rows selected
SQL>
SQL> begin atomic
cont> call p2 ();
cont> signal 'ERROR';
cont> end;
%RDB-E-SIGNAL_SQLSTATE, routine "(unnamed)" signaled SQLSTATE "ERROR"
SQL>
SQL> -- there should still be no rows
SQL> -- due to the failed ATOMIC block
SQL> select * from t1;
          F1
           1
1 row selected
SQL>
```

The only workaround is to make the nested stored procedure also use ATOMIC compound statements.

This problem has been corrected in Oracle Rdb Release 7.0.6.2. The ATOMIC state of the transaction is now correctly seen by all nested stored procedures.

4.2.4 GROUP BY Queries Fail With INVALID_BLR Error

Bug 1757309

In prior releases of Oracle Rdb 7.0, it was possible to get a INVALID_BLR error when processing a SELECT expression that used GROUP BY on an expression.

The following example shows the type of query and the resulting error.

```
SQL> select last_name || first_name,
cont>   count( last_name || first_name)
cont> from employees
cont> group by last_name ||first_name;
%RDB-E-INVALID_BLR, request BLR is incorrect at offset 91
```

This problem is caused by the query processor matching the value expression (the concatenate of LAST_NAME and FIRST_NAME in the example) within the aggregate function. This matching should not be performed within aggregate functions (COUNT, MAX, MIN, AVG, and SUM) since these expressions are

filters on the contributing rows of the group and not references to the final result.

This problem has been corrected in Oracle Rdb Release 7.0.6.2.

4.2.5 Using Null Indicators With Dynamic SQL and Compound Statements Yields Incorrect Results

Bug 1394684

When using compound statements in Dynamic SQL, an attempt to set the null indicator produced wrong results. Specifically, the parameter marker's column(s) would not be set to null.

An example follows:

```
SQL> create database filename test
SQL> create table test_table (fld1 char (10));
SQL> commit;
SQL> exit;

! A sample C program using Dynamic Sql, parameter markers, and setting the
! null indicator bit.
#include <stdio.h>
#include <string.h>

#define MAX_PARAMS 1

struct SQLDA_STRUCT
{
        char SQLDAID[8];
        int SQLDABC;
        short SQLN;
        short SQLD;
        struct
        {
                short SQLTYPE;
                short SQLLEN;
                char *SQLDATA;
                int *SQLIND;
                short SQLNAME_LEN;
                char SQLNAME[30];
        } SQLVAR[MAX_PARAMS];
} *PARAM_SQLDA;

main()
{
        long SQLCODE;
        char sql_statement[256];
        char insert_param[11];
        int insert_indicator;

        PARAM_SQLDA = malloc((MAX_PARAMS * 44) + 16);
        PARAM_SQLDA->SQLN = MAX_PARAMS;

        strcpy(sql_statement, "attach 'filename test'");

        exec sql execute immediate :sql_statement;

 /*
 * Insert within an MSP - Null was not inserted correctly
 */
strcpy(sql_statement,"begin ");
```

```
strcpy(sql_statement,"insert into test_table (fld1) values (?); ");
strcpy(sql_statement,"end");

 /* Prepare the statement */

exec sql prepare sql_statement_id from :sql_statement;

insert_indicator = -1;

PARAM_SQLDA->SQLVAR[0].SQLDATA = insert_param;
PARAM_SQLDA->SQLVAR[0].SQLIND = &insert_indicator;

 /* Execute the statement */

exec sql execute sql_statement_id using descriptor PARAM_SQLDA;

! Selecting from the table shows incorrect results.
select * from test_table;
 FLD1
 ..........
1 row selected
   .
   .
   .
```

This problem has been corrected in Oracle Rdb Release 7.0.6.2. When using a compound statement in Dynamic SQL, assigning "-1" to the indicator field (SQLIND) in the SQLDA now sets the associated column to NULL.

The following example shows the correct results:

```
select * from test_table;
 FLD1
 NULL
1 row selected
   .
   .
   .
```

4.2.6 Using the CALL Statement in Dynamic SQL Results in Input Parameters Not Being Written to SQLDA

Bug 1324098

When using the CALL statement to invoke a stored procedure in Dynamic SQL, input parameters were not written to the SQLDA. In addition, parameters which were defined without an access mode, that is, without IN, OUT, or INOUT, were also not written to the SQLDA.

The problem did not occur when using the MSP (multistatement procedure) CALL statement; that is the CALL statement within a BEGIN...END statement.

The following example uses interactive SQL to create a MODULE and then CALL the stored procedure. This works properly.

```
SQL>create data file test;
SQL> create module test_module language sql
cont> procedure test_param (:x int, :test_string char(32), out :status int);
cont> begin
```

```
cont> set :status = 4;
cont> end;
cont> end module;
SQL> commit;
SQL> declare :s_status int;
SQL> declare :v_user char (32);
SQL> declare :xx int;
SQL> begin
cont> set :v_user = 'TESTUSER';
cont> SET :xx = 123;
cont> call test_param (:xx, :v_user, :s_status);
cont> end;
SQL> print :s_status;
    S_STATUS
           4
SQL> exit
```

The next example uses a Dynamic program to attach to the database and then CALL the stored procedure. This is an internal program which, when run, shows input and output parameters written to the SQLDA.

The first example illustrates correct behavior using the MSP CALL statement. The second example illustrates incorrect behavior using the simple CALL statement. Note that, in the second example, there are no IN parameters written to the SQLDA (ie. there is no IN:)

Example 1:

```
$ run dyntest
Enter statement:
attach 'filename test';
Enter statement:
begin call test_param (?,?,?); end;
in: 0:    10001
in: 1:    20001
out: 0:    30005
```

Example 2:

```
Enter statement:
call test_param (?,?,?);
out: 0:    30005
```

The problem has been fixed. Both input parameters as well as parameters defined without an access mode are now correctly written to the SQLDA.

```
Enter statement:
call test_param (?,?,?);
in: 0:    10001
in: 1:    20001
out: 0:    30005
```

The following example illustrates the error message generated when using Powerbuilder. This error message should no longer occur.

```
string ls_string
long ll_zahl

ll_zahl = 55
ls_string = 'EGE'
DECLARE test procedure for test_param :ll_zahl, :ls_string, output using sqlca;

long ll_status
```

```
//Execute
ll_status = −1
execute test ;
if sqlca.sqlcode <> 0 then
        messagebox ('execute', 'Error' + sqlca.sqlerrtext)
end if
fetch test into :ll_status;
commit;

sle_1.text = string (ll_status)

ODBC − Error Msg

S1093 Invalid Parameter Number
```

This problem has been corrected in Oracle Rdb Release 7.0.6.2.

4.2.7 Incorrect Processing of CASE Expression

Bug 850442

In prior releases of Oracle Rdb, the SQL interface incorrectly processed CASE expressions which included statistical functions (i.e. COUNT, MAX, MIN, AVG and SUM).

The following example, which imbeds statistical functions in a CASE expression, caused Rdb to bugcheck:

```
select
    case
        when count(employee_id) >= 1
            then '1'
        when count(employee_id) = 0
            then '2'
        else '3'
    end
 from employees;
%RDMS−I−BUGCHKDMP, generating bugcheck dump file USER2:[TEST]RDSBUGCHK.DMP;
%SQL−I−BUGCHKDMP, generating bugcheck dump file USER2:[TEST]SQLBUGCHK.DMP;
%SYSTEM−F−ACCVIO, access violation, reason mask=00, virtual
address=0000000000000098, PC=000000000038B948, PS=0000001B
```

This problem has been corrected in Oracle Rdb Release 7.0.6.2.

This improved handling of statistical functions also corrects some query strategies. The following example implements a simple ABS functionality. Due to the erroneous handling of the statistical function, an extra subselect was present as shown in the optimizer STRATEGY display.

```
SQL> set flags 'strategy';
SQL> select
cont>    case
cont>        when AVG (salary_amount) < 0 then − AVG (salary_amount)
cont>        else AVG (salary_amount)
cont>    end
cont> from SALARY_HISTORY;
Cross block of 2 entries
  Cross block entry 1
    Aggregate       Get     Retrieval sequentially of relation SALARY_HISTORY
  Cross block entry 2
    Aggregate       Get     Retrieval sequentially of relation SALARY_HISTORY

  2.652896707818930E+004
1 row selected
```

The corrected SQL query now only requires a single table access.

```
Aggregate       Get     Retrieval sequentially of relation SALARY_HISTORY

  2.652896707818930E+004
1 row selected
```

Applications that encounter this type of unexpected optimizer strategy will need to be recompiled and stored procedures and functions will need to be recreated.

4.2.8 %RDB−E−NO_DIST_BATCH_U Error When Executing SET TRANSACTION

Bug 1921672

If a SET TRANSACTION statement was executed to start a distributed transaction (2 phase commit) and which specified certain table partitions, an error was inappropriately returned. Specifically, if partition 14 was named, Rdb would return a %RDB−E−NO_DIST_BATCH_U error and not start the transaction. For example, suppose an Interactive SQL session has two databases attached (this implicitly starts a DECdtm distributed transaction), the following SQL commands would fail as shown:

```
SET TRANSACTION READ WRITE WAIT ISOLATION LEVEL READ COMMITTED −
RESERVING DB2.MY_TABLE PARTITION(14) FOR EXCLUSIVE WRITE;
%RDB−E−NO_DIST_BATCH_U, no distributed transaction is allowed with the
recovery mechanism disabled
```

This query will now execute normally and start a distributed transaction.

This problem has been corrected in Oracle Rdb Release 7.0.6.2.

4.3 Oracle RMU Errors Fixed

4.3.1 RMU/Extract Fails to Extract IMPORT Script from Multischema Database

RMU/Extract would produce an error when attempting to extract an IMPORT script from a multischema database. The following example shows the error which occurs from this attempt.

```
$ rmu/extract/item=import sample.rdb
-- RMU/EXTRACT for Oracle Rdb V7.0-4       16-FEB-2001 08:39:25.25
--
--                   Physical Database Definition
--
-----------------------------------------------------------------
import database from rmuextract_rbr
    filename 'DEVICE:[DATABASE]SAMPLE.RDB'
    protection is ACL
    multischema is ON
%RDB-E-STREAM_EOF, attempt to fetch past end of record stream
%RMU-F-FATALRDB, Fatal error while accessing Oracle Rdb.
%RMU-F-FTL_RMU, Fatal error for RMU operation at 16-FEB-2001 08:39:27.71
```

This problem has been corrected in Oracle Rdb Release 7.0.6.2. The only workaround is to edit the output from the /ITEM=DATABASE command which has a similar format.

4.3.2 RMU/Extract May Abort with ACCVIO and Bugcheck

Bugs 1314250, 1368926, 638001, 1644617

In prior releases of Oracle Rdb on OpenVMS Alpha, RMU/Extract may abort with an unexpected error and generate a bugcheck. This is shown in the following example:

```
$ rmu/extract/item=all/out=t.t dba_database
%COSI-F-UNEXPERR, unexpected system error
-SYSTEM-F-ACCVIO, access violation, reason mask=8C, virtual
address=0000000000000001B, PC=0000000000000003, PS=7AD99FF8
%RMU-F-FATALOSI, Fatal error from the Operating System Interface.
%RMU-F-FTL_RMU, Fatal error for RMU operation at 15-FEB-2001 11:13:15.93
```

Oracle believes this is related to a flaw in the OpenVMS runtime library routine LIB$CVT_DX_DX in use by RMU/Extract. Access to an unaligned source buffer is the most likely cause of this error.

This problem has been corrected in Oracle Rdb Release 7.0.6.2. Rdb now uses a quadword aligned buffer for conversions. A workaround for the current Rdb releases may be to upgrade to a corrected version of OpenVMS. Please contact COMPAQ Customer Support for further details.

4.3.3 RMU/Extract /ITEM=WORKLOAD Generates Incomplete Output

In prior releases of Oracle Rdb, the RMU/Extract /ITEM=WORKLOAD command would cause an incomplete script to be generated. This occurred when the workload column group consisted of more than one column.

This problem has been corrected in Oracle Rdb Release 7.0.6.2.

4.3.4 RMU Commands May Leave Zero Length Log File

In prior releases of Oracle Rdb, several RMU commands would open the file specified by /LOG prior to completing checks on other qualifiers. This could cause a zero length file to be created by a failing RMU command.

These commands have been corrected: RMU Collect Optimizer_Statistics, RMU Delete Optimizer_Statistics, RMU Insert Optimizer_Statistics, and RMU Show Optimizer_Statistics.

This problem has been corrected in Oracle Rdb Release 7.0.6.2.

4.3.5 %RDB−E−INVALID_BLR When Group By Count(*)

Bug 1484666

The COUNT() aggregate function may not be used in the GROUP BY clause of a SELECT statement, but a query containing the COUNT() function in a GROUP_BY clause returned an inappropriate error message as shown in the following example:

```
SQL> select count(*) from employees
cont> group by count(*);
%RDB−E−INVALID_BLR, request BLR is incorrect at offset 44
```

This query will now return the following error message:

```
%SQL−F−INVFUNREF, Invalid function reference
```

This problem has been corrected in Oracle Rdb Release 7.0.6.2.

4.3.6 Full Logical Area Name Displayed in Zoom Screen

In previous versions of Oracle Rdb, RMU /SHOW STATISTICS was unable to display the full logical area name and storage area name in the Logical Area Overview "View" screen if the length of the logical area and storage area exceeded approximately 40 characters.

This problem has been corrected in Oracle Rdb Release 7.0.6.2. RMU /SHOW STATISTICS now displays the Logical Area Overview "View" screen with the right edge of the view screen tied to the width of the display. Setting the display wider (to 132 columns, for example) allows the full logical area name and storage area name to be displayed.

4.3.7 RMU /UNLOAD Specifying Both /VIRTUAL and /RECORD_DEFINITION

Previously, the RMU /UNLOAD command qualifier VIRTUAL_FIELDS could only be specified if the RECORD_DEFINITION qualifier was also specified. This restriction prevented unloading certain classes of data (database keys within views, for example) into the specially structured file format that contains both the data and the metadata (.unl).

This problem has been corrected in Oracle Rdb Release 7.0.6.2. The VIRTUAL_FIELDS qualifier is now permitted regardless of the setting of the RECORD_DEFINITION qualifier.

4.3.8 RMU /SET AFTER_JOURNAL /SWITCH and Automatic Backup Server Does Not Backup All Journals

Bug 1614198

When using the Automatic Backup Server (ABS) process, if multiple after−image journal files were eligible for backup, it was possible that not all journals would be backed up when the next ABS was started as a result of an after image journal switch. This problem would be most apparent when the ABS was suspended through multiple AIJ switch operations.

This problem has been corrected in Oracle Rdb Release 7.0.6.2. During an after image journal switch operation, one ABS process is started for each AIJ file that needs to be backed up. These ABS processes operate in parallel with each other performing a backup from one AIJ file to a backup AIJ file based on the backup file specification. This represents a difference in behavior from prior versions where one ABS would backup all eligible AIJ files to a single AIJ backup file regardless of the setting of the backup file specification.

4.3.9 RMU /VERIFY Constraint Reports Erroneous Error

Bug 1575187

The RMU /VERIFY of constraints on OpenVMS Alpha could sometimes cause a constraint violation error when the data was in fact valid. This is shown in the following example.

```
$ rmu/verify/constraint constr_test
%RMU-I-CONSTFAIL, Verification of constraint "LI_CONSTRAINT" has failed.
```

Only constraints with expressions such as CASE would fail for this reason. Verification of this type of constraint would always fail on OpenVMS Alpha.

Verification of the same constraint on OpenVMS VAX would succeed.

This problem has been corrected in Oracle Rdb Release 7.0.6.2.

4.3.10 RMU/DUMP/AFTER /START and /END Qualifiers are Difficult to Use

The /START and /END qualifiers for the RMU/DUMP/AFTER_JOURNAL utility can be difficult to use as users seldom know, nor can they determine, the AIJ record number in advance of using the utility.

There is no workaround to this problem.

This problem has been corrected in Oracle Rdb Release 7.0.6.2. The RMU/DUMP/AFTER_JOURNAL utility 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" 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.
- TYPE=(aijtype) : Specifies the AIJ journal record types to dump. One or more "aijtype" keywords may be specified. Valid "aijtype" keywords are:
  - ♦ ACE_HEADER − TYPE=A records
  - ♦ CHECKPOINT − TYPE=B records
  - ♦ CLOSE − TYPE=K records
  - ♦ COMMIT − TYPE=C records
  - ♦ DATA − TYPE=D records
  - ♦ GROUP − TYPE=G records
  - ♦ INFORMATION − TYPE=N records
  - ♦ OPEN − TYPE=O records
  - ♦ OPTIMIZE_INFORMATION − TYPE=I records
  - ♦ PREPARE − TYPE=V records
  - ♦ ROLLBACK − TYPE=R records

  This keyword can only 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.

4.3.11 RMU/LOAD FILACCERR Exception While Reading Input File

Previously, if an error occurred while reading an input file, the RMU /LOAD utility would signal a

FILACCERR exception along with the RMS "STS" value. However, the RMS "STV" field was not displayed thus limiting some information about the actual cause of the failure.

This problem has been corrected in Oracle Rdb Release 7.0.6.2. RMU /LOAD now displays both the RMS "STS" and "STV" values during a file read failure condition.

4.3.12 RMU/LOAD Access Violation When Table Constraints Were Defined

An access violation occurred when RMU/LOAD was loading data in a table with table constraints defined.

The following example shows that an access violation occurred in RMU/LOAD when table constraints were defined for the table being loaded.

```
$ RMU/LOAD DBA_DATABASE EMP EMP.UNL

%SYSTEM−F−ACCVIO, access violation, reason mask=00, virtual

address=00000000026EDFF1, PC=FFFFFFFF8089D778, PS=0000001B

%RMU−I−BUGCHKDMP, generating bugcheck dump file

DB_USER:[USER]RMUBUGCHK.DMP

%RMU−I−DATRECSTO,  0 data records stored.

%RMU−F−FTL_LOAD, Fatal error for LOAD operation at 16−JAN−2001 16:20:07.33
```

As a workaround to this problem, specify /NOCONSTRAINTS when doing the RMU/LOAD.

```
RMU/LOAD/NOCONSTRAINTS DBA_DATABASE EMP EMP.UNL
```

This problem has been corrected in Oracle Rdb Release 7.0.6.2.

4.3.13 RMU/UNLOAD/AFTER_JOURNAL CPU Loop With Large Fragmented Record

Previously, it was possible for a row fragment larger than approximately 1500 bytes to cause the RMU /UNLOAD /AFTER_JOURNAL command to become wedged in a CPU loop. The RMU/UNLOAD/AFTER_JOURNAL utility was incorrectly processing this type of record in some cases and could continuously search an internal queue while looking for a non−existent row fragment.

This problem has been resolved in Oracle Rdb Release 7.0.6.2. The maximum size of an internal sort buffer has been increased to allow for any possible fragmented record size. In addition, an estimate is made of the largest likely record size to be extracted and this size is used as a "hint" while calculating the record sort size. Finally, an additional sanity check has been included to prevent an endless loop while searching for fragments. When a fragment is not found within a transaction, the RMU/UNLOAD/AFTER_JOURNAL utility will now cause a bugcheck dump rather than looping indefinitely.

4.3.14 RMU/VERIFY/INDEX/TRANS=READ_ONLY Did Not Detect BADIDXREL

Bug 1773334

When RMU/VERIFY/INDEX/TRANSACTION=READ_ONLY verified an index with a bad pointer to a data record, it failed to detect the problem and to output the RMU−W−BADIDXREL message. If /TRANSACTION=PROTECTED (the default transaction mode) or /TRANSACTION=EXCLUSIVE was specified, the problem was detected and the RMU−W−BADIDXREL message was output. This problem has been corrected and if RMU/VERIFY/INDEX/TRANSACTION=READ_ONLY is specified, the bad index data record pointer will be detected and the RMU−W−BADIDXREL message will be output.

The following example shows that RMU/VERIFY/INDEX/TRANSACTION=READ_ONLY failed to detect the bad record pointer in the TABLE_1_INDEX index, but if the transaction mode specified was PROTECTED (the default) or EXCLUSIVE, the bad index pointer was detected and the RMU−W−BADIDXREL message was output.

```
$RMU/VERIFY/INDEX/TRANSACTION=READ_ONLY BADINDEX.RDB

$RMU/VERIFY/INDEX BADINDEX.RDB
%RMU-W-BADIDXREL, Index TABLE_1_INDEX either points to a non-existent record
or has multiple pointers to a record in table TABLE_1.
The logical dbkey in the index is 121:1894:10.

$RMU/VERIFY/INDEX/TRANSACTION=PROTECTED BADINDEX.RDB
%RMU-W-BADIDXREL, Index TABLE_1_INDEX either points to a non-existent record
or has multiple pointers to a record in table TABLE_1.
The logical dbkey in the index is 121:1894:10.

$RMU/VERIFY/INDEX/TRANSACTION=EXCLUSIVE BADINDEX.RDB
%RMU-W-BADIDXREL, Index TABLE_1_INDEX either points to a non-existent record
or has multiple pointers to a record in table TABLE_1.
The logical dbkey in the index is 121:1894:10.
```

As a workaround to this problem, do not specify /TRANSACTION=READ_ONLY with the RMU/VERIFY/INDEX command.

```
$RMU/VERIFY/INDEX BADINDEX.RDB
%RMU-W-BADIDXREL, Index TABLE_1_INDEX either points to a non-existent record
or has multiple pointers to a record in table TABLE_1.
The logical dbkey in the index is 121:1894:10.
```

This problem has been corrected in Oracle Rdb Release 7.0.6.2.

4.3.15 RMU /UNLOAD Closes .RRD File Earlier

Previously, the RMU /UNLOAD command did not close the generated record definition (.RRD) file until the end of the entire unload operation. This could be inconvenient when passing the record definition to another application through, for example, an OpenVMS mailbox.

The RMU /UNLOAD command now closes the .RRD file as soon as it has been written. This allows other utilities to read the .RRD file as soon as it has been created and written.

This problem has been corrected in Oracle Rdb Release 7.0.6.2.

4.3.16 RMU /UNLOAD /AFTER_JOURNAL Requires Accurate AIP Logical Area Information

The RMU /UNLOAD /AFTER_JOURNAL command uses the on−disk area inventory pages (AIPs) to determine the appropriate "type" of each logical area when reconstructing logical DBKEYs for records stored in mixed−format storage areas. However, the logical area "type" information in the AIP is generally

"unknown" for logical areas created prior to Oracle Rdb V7.0.1. If the RMU /UNLOAD /AFTER_JOURNAL command cannot determine the logical area type for one or more AIP entries, a warning message is displayed for each such area and may ultimately return logical DBKEYs with a "0" (zero) area number for records stored in mixed−format storage areas.

In order to update the on−disk logical area type in the AIP, the RMU /REPAIR utility must be used. The "/INITIALIZE = LAREA_PARAMETERS = (optionfile)" qualifier option file can be used with the "/TYPE" qualifier. For example, to repair the EMPLOYEES table of the MF_PERSONNEL database, you would create an options file that contains the following line:

```
EMPLOYEES /TYPE=TABLE
```

For partitioned logical areas, the /AREA=name qualifier can be used to identify the specific storage areas that are to be updated. For example, to repair the EMPLOYEES table of the MF_PERSONNEL database for the EMPID_OVER storage area only, you would create an options file that contains the following line:

```
EMPLOYEES /AREA=EMPID_OVER /TYPE=TABLE
```

The /TYPE qualifier specifies the type of a logical area. The following keywords are allowed:

Note

This type should NOT be used for the RDB$SYSTEM logical areas. This type does NOT identify system relations.

Table 4−1 Valid / TYPE keywords

Keyword	Meaning
TABLE	Specifies that the logical area is a data table. This would be a table created using the SQL "CREATE TABLE" syntax.
BTREE	Specifies that the logical area is a b−tree index. This would be an index created using the SQL "CREATE INDEX TYPE IS SORTED" syntax.
HASH	Specifies that the logical area is a hash index. This would be an index created using the SQL "CREATE INDEX TYPE IS HASHED" syntax.
SYSTEM	Specifies that the logical area is a system record which is used to identify hash buckets. Users cannot explicitly create these types of logical areas.
BLOB	Specifies that the logical area is a blob repository.

There is no explicit error checking of the "type" specified for a logical area. However, an incorrect type may cause the RMU /UNLOAD /AFTER_JOURNAL command to be unable to correctly return valid logical DBKEYs.

4.3.17 Asterisks Displayed for STID on >99 Attaches in RMU/SHOW STATISTICS

Bug 1704207

When the streamid (STID) increases beyond 99, display of the PID:STID in the RMU/SHOW Statistics screens is faulty. STID >99 is not displayed correctly unless one zooms in.

This problem has been corrected in Oracle Rdb Release 7.0.6.2. Now "**" are shown in RMU/SHOW statistics screens when the STID is >99. The user can zoom in and see the actual value.

4.3.18 RMU/SHOW STATISTICS Displays Physical Area Name for Page Lock

When possible, the RMU/SHOW STATISTICS "LockID" pop−up screen now displays the physical area name of the storage area along with the area number for page locks.

The following example shows the storage area name (RDB$SYSTEM) being displayed.

```
+-------------------------------------------------------------+
|                                                             |
|            Resource: page 3272 (area 1 RDB$SYSTEM)          |
| State... Process.ID Process.name... Lock.ID. Rq Gr Queue    |
|                                                             |
| Blocker: 21E74357   njl @ TNA455    660100A7    EX Grant    |
| Waiting: 21E6E356   njl @ TNA454    5C00DCFC CR    Wait     |
|                                                             |
+-------------------------------------------------------------+
```

This problem has been corrected in Oracle Rdb Release 7.0.6.2.

4.3.19 RMU/SHOW STATISTICS Incorrect AIJ CurrEOF Value

Previously, the RMU/SHOW STATISTICS Utility would sometimes display an incorrect value of 2 for the current AIJ file EOF. This could happen when the AIJ end−of−file location had not been established on the current node.

Now, when the AIJ end−of−file location is not accurately known, the RMU/SHOW STATISTICS utility indicates "Unknown" for the CurrEOF column of the "AIJ Journal Information " screen as shown in the following example.

```
Node: SLAM (1/1/1) Oracle Rdb X7.0−00 Perf. Monitor 26−JUN−2001 10:11:08.16
Rate: 3.00 Seconds     AIJ Journal Information         Elapsed: 00:03:30.91
Page: 1 of 1                DUA0:[DB]DB.RDB      Mode: Online
−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
Journaling: enabled   Shutdown:   60  Notify: disabled  State: Accessible
ALS: Manual     ABS: disabled  ACE: disabled  FC: enabled   CTJ: disabled
ARB.Count:  300 ARB.Avail:  300 SwtchSched:  0 NxtSwtch:
After−Image.Journal.Name....... SeqNum AIJsize CurrEOF Status. State......
J1                              Unused    5000   Empty Latent  Accessible
J2                                  11    5000 Unknown Current Accessible
J3                              Unused    5000   Empty Latent  Accessible
J4                              Unused    5000   Empty Latent  Accessible
J5                              Unused    5000   Empty Latent  Accessible
```

This problem has been corrected in Oracle Rdb Release 7.0.6.2.

4.3.20 RMU /UNLOAD /AFTER_JOURNAL Excessive Work File I/O

Previously, the RMU /UNLOAD /AFTER_JOURNAL Utility would access work files more often than needed. This additional disk I/O could greatly increase the amount of time required to extract certain classes of AIJ files.

Several performance enhancements have been included to reduce disk I/O by buffering and sorting additional information in memory. Previously, up to 512 records would be sorted using an internal sort algorithm (thus avoiding calling the VMS SORT32 package). This threshold has been increased to 5000 records. The

per−transaction work file memory buffer size has also been significantly increased. These changes may require additional process working set to avoid excessive page faulting.

This problem has been corrected in Oracle Rdb Release 7.0.6.2.

4.3.21 RMU/Extract Not Formatting View Column Expressions Correctly

Bug 1832240

In prior releases of Oracle Rdb, the RMU/Extract command did not correctly format VIEW definitions that contained computed expressions in the SELECT clause, such as that shown below.

```
create view V1 (F3) as
  select sum (F3 /
              case (select cast (F1 as integer) from T1
                     where F2 = 'STR_VALUE')
               when 0 then 1
               when 1 then 10
               when 2 then 100
               when 3 then 1000
               when 4 then 10000
               when 5 then 100000
               else 0
              end)
    from T2;
```

This example was extracted and the results are shown below: note the lack of formatting in the expression and the missing separating white space. This made the generated definition illegal.

```
create view "V1"
    (F3) as
    select

     sum((C2.F3 / case (select CAST(C3.F1 AS INTEGER) from T1 C3where (C3.F2 =
    'STR_VALUE')) when 0 then 1 when 1 then 10 when 2 then 100 when 3 then 1000
     when 4 then 10000 when 5 then 100000 else 0end)) from T2 C2;
```

The only workaround for this problem is to manually edit the definition after extracting with RMU/Extract or revert to the original view source.

This problem has been corrected in Oracle Rdb Release 7.0.6.2.

4.3.22 Recovery Journals With Only Rollback Records Not Handled Correctly

Bug 1544303

When applying a roll forward journal which contained only a rollback record, the highest TSN in the database was incorrectly set in the TSN block. As a result, the first committed record in the next roll forward journal was being ignored, since the recover code did not consider the next transaction applied to the current recovery.

There is no workaround to this problem, other than to try and avoid journals which contain only the rollback transaction.

This problem has been corrected in Oracle Rdb Release 7.0.6.2.

4.3.23 RMU/UNLOAD/AFTER_JOURNAL Fragmented Records Clarification

The RMU /UNLOAD /AFTER_JOURNAL Utility uses additional CPU and memory resources while processing and unloading fragmented records from the after−image journal file. As record fragments are found within a transaction, they are buffered, in memory, on a "fragment" queue. After all non−fragmented records from the transaction have been output, the fragmented records are reconstructed and output.

Because the fragments are buffered in memory, additional process page file quota may be required when unloading transactions that have a large number of record fragments. Also, additional process working set quota may be required in order to limit process page faulting.

Chapter 5
Software Errors Fixed in Oracle Rdb Release 7.0.6.1

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

5.1 Software Errors Fixed That Apply to All Interfaces

5.1.1 Excessive Pages Checked/Discarded When Storing New Rows

Bug 1391003

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

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

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

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

5.1.2 Quota Exceeded Conditions During DAPF May Lead To Missing Updates

Bug 1485622

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

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

Further, accounts and processes that access Oracle Rdb databases should be reviewed to ensure that various quotas are set to ensure high levels of I/O performance. Table 5−1 lists suggested quota values for maximum performance of Rdb I/O operations.

Table 5−1 Recommended Minimum Process Quotas

Quota	Setting
DIOLM	Equal to or greater than the count of database buffers (either the database default or the setting of the logical name RDM$BIND_BUFFERS) when local buffers are enabled for a database or a value greater than the global buffer USER LIMIT setting. Minimum of 250.
BIOLM	Equal to or greater than the setting of DIOLM.
ASTLM	Equal to or greater than 50 more than the setting of DIOLM.

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

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

5.1.3 Bugcheck at LCKCCH$LOCK_RET_NOT_OK During Hash Index Creation

Bug 1430433

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

```
$ DEFINE RDMS$CREATE_LAREA_NOLOGGING "1"
$ SQL$
SQL> attach data file 'DUA0:[DB]DB';
SQL> declare transaction read write reserving T1 for exclusive write;
SQL>
SQL> create unique index T1I on T1 (C1, C2, C3) type is HASHED store in A1;
%RDB-W-META_WARN, metadata successfully updated with the reported warning
-RDMS-W-DATACMIT, unjournaled changes made; database may not be recoverable
-RDMS-W-DOFULLBCK, full database backup should be done to ensure future recovery
SQL>
SQL> create index T2I on T2 (C1) type is HASHED store in A2;
%RDMS-I-BUGCHKDMP, generating bugcheck dump file DUA0:[DB]RDSBUGCHK.DMP;
```

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

```
$ SEARCH RDSBUGCHK.DMP "EXCEPTION","SAVED PC","-F-","-E-"
***** Exception at 00924930 : LCKCCH$LOCK_RET_NOT_OK + 000000F0
%RDMS-F-NOT_LARDY, area for 54:2:0 not in proper ready mode
Saved PC = 00894358 : DIOFETCH$FETCH_ONE_LINE + 00000208
Saved PC = 00895088 : DIO$FETCH_DBKEY + 000002D8
Saved PC = 00952DC0 : PSI$MODIFY_PCL_ALL + 00000090
Saved PC = 0033E2B4 : SOR$$GET_KEY_PREFIX + 00000A84
Saved PC = 0033E898 : DIODROPDROPMIXEDAREA + 00000308
Saved PC = 008B3330 : DIOLAREA$ERASE_MIXED_LAREA + 00000210
Saved PC = 008B0ED0 : DIOLAREA$DELETE_LAREA + 00000180
Saved PC = 009AB35C : DIOUN$CRLA + 0000057C
Saved PC = 009A9D60 : DIO$UN_DO + 000001C0
Saved PC = 0097661C : RUJUTL$ROLLBACK_LOOP + 000004CC
Saved PC = 00973700 : RUJ$VERB_ROLLBACK + 00000080
Saved PC = 008D84F0 : KOD$VERB_FAILURE + 000000D0
Saved PC = 006C7688 : RDMS$$TOP_DSDI_CLEANUP + 000000A8
Saved PC = 006C70E4 : RDMS$$TOP_DSDI_HNDLR + 00000174
Saved PC = 808A3D94 : symbol not found
```

```
Saved PC = 958746BC : symbol not found
***** Exception at 006A83C0 : RDMS$$KOD_INT_READY + 000001D0
%RDMS-F-NOT_LARDY, area for 54:2:0 not in proper ready mode
Saved PC = 006F7748 : RDMS$$EXE_OPEN + 00000E28
Saved PC = 006F6AA0 : RDMS$$EXE_OPEN + 00000180
Saved PC = 006F69D0 : RDMS$$EXE_OPEN + 000000B0
Saved PC = 006F82B4 : RDMS$$EXE_OPEN + 00001994
Saved PC = 0466EAB4 : symbol not found
Saved PC = 0082C688 : RDMS$$INT_START_REQUEST + 00000098
Saved PC = 00470818 : RDMS$$COUNT_RELATION + 000001E8
Saved PC = 004D56A4 : RDMS$$CREATE_INDEX_INFO + 00003F74
Saved PC = 004596B8 : RDMS$$RELEASE_DDL_VM_HNDLR + 00001058
Saved PC = 003C25A4 : BLI$CALLG + 000000BC
Saved PC = 008D55F0 : KOD$SETSTK_AND_CONTINUE + 00000184
```

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

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

```
SQL> declare transaction read write reserving T1 for exclusive write;
SQL>
SQL> create unique index T1I on T1 (C1, C2, C3) type is HASHED store in A1;
%RDB-W-META_WARN, metadata successfully updated with the reported warning
-RDMS-W-DATACMIT, unjournaled changes made; database may not be recoverable
-RDMS-W-DOFULLBCK, full database backup should be done to ensure future recovery
SQL>
SQL> create index T2I on T2 (C1) type is HASHED store in A2;
%RDB-E-NO_META_UPDATE, metadata update failed
-RDB-E-UNRES_REL, relation T2 in specified request is not a relation reserved
  in specified transaction
SQL>
```

5.1.4 Attempts to Truncate Snapshot Files Online Hang

Bug 563410

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

1. Session 1: Start a read only transaction

   ```
   SQL> attach 'file mf_personnel';
   SQL> set transaction read only reserving jobs for shared read;
   SQL> select * from jobs;
   ```
2. Session 2: Start another read only transaction

   ```
   SQL> attach 'file mf_personnel';
   SQL> set transaction read only reserving jobs for shared read;
   SQL> select * from jobs;
   ```
3. Session 3: Attempt to truncate a snapshot file

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

4. Session 1: Start another read only transaction

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

5. Session 2: Start another read only transaction

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

.
.
.

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

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

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

Bug 754173

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

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

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

5.1.6 Date Function Causes RDML/PASCAL Compilation Problems

Bug 908356

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

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

```
{== bug.rpa ============================}
PROGRAM Personnel (input, output);
DATABASE PERSONNEL = FILENAME 'mf_personnel';
TYPE
```

```
    dummy_rec = PACKED RECORD
                    salary_start  : rdml$cddadt_type;
                END;
    dummy_ptr = [unsafe] ^dummy_rec;
VAR
    dum_ptr : dummy_ptr;

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


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

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

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

Bug 1477955

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

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

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

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

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

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

```
        COMMIT;
    END.

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

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

There is no workaround for this problem.

These problems have been corrected in Oracle Rdb Release 7.0.6.1.

5.1.7 RDBPRE Results in MAXARGEXC Warning from Alpha MACRO Compiler

Bug 1111805

In prior releases of Oracle Rdb 7.0, the RDBPRE pre−processor may generate intermediate code which causes the Alpha MACRO compiler to generate a warning during compilation. For example,

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

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

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

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

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

&Rdb& start_stream TEST_STREAM

&Rdb& fetch TEST_STREAM

&Rdb& end_stream TEST_STREAM

END PROGRAM
```

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

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

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

5.1.8 Error Writing File SORTWORK.TMP, Normal Successful Completion

Bug 1495500

Occasionally, a query would fail with the following errors:

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

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

This problem has been corrected in Oracle Rdb Release 7.0.6.1.

5.1.9 Extraneous Logical Area Created by DROP STORAGE MAP

Bug 703265

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

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

This problem has been corrected in Oracle Rdb Release 7.0.6.1.

5.1.10 Cannot Disable SAME BACKUP FILENAME Clause

Bug 1240826

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

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

5.1.11 Query Having OR Compound Predicates With Subquery Returns Wrong Results

Bug 1527102

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

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

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

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

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

This problem has been corrected in Oracle Rdb Release 7.0.6.1.

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

Bug 1569972

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

```
set flags 'strategy,detail';

select t1.c1 from t1 t1, t2 t2 where
((t2.c4 = 1 and
  t2.c5 = 5 and
  not exists (select * from t2 t2a                              <----
              where t2a.c4 = 4 and t2a.c5  = 5)) or             <----
 (t2.c4  = 4 and t2.c5 = 5))                                    <----
 and t1.c1 = t2.c6                                              <----
;
Tables:
  0 = T1
  1 = T2
  2 = T2
```

```
Cross block of 3 entries
  Cross block entry 1
    Conjunct: {subselect} = 0
      Aggregate-F1: (COUNT-ANY)     Index only retrieval of relation 2:T2
      Index name  T2_H [2:2]
        Key: (2.C4 = 4) AND (2.C5 = 5)
  Cross block entry 2
    Conjunct: (1.C4 = 1) OR (1.C4 = 4)
    Conjunct: 1.C5 = 5
    Conjunct: {subselect} = 0
    Get     Retrieval by index of relation 1:T2
      Index name  T2_H [(2:2)2] Bool
        Key: ((1.C4 = 1) AND (1.C5 = 5)) OR ((1.C4 = 4) AND (1.C5 = 5))
        Bool: 1.C5 = 5
  Cross block entry 3
    Index only retrieval of relation 0:T1
      Index name  T1_H [1:1]
        Key: 0.C1 = 1.C6
0 rows selected
```

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

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

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

```
set flags 'strategy,detail';

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

This problem has been corrected in Oracle Rdb Release 7.0.6.1.

5.1.13 Query Using German Collating Sequence Returns Wrong Results

Bug 1530947

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

```
SELECT  p.datum, p.produkt, p.abtlg, p.stelle
 FROM v_team_datum p,
      produkte g
 where
       p.abtlg=g.abtlg  ;
Conjunct
Match
  Outer loop
    Sort    Conjunct        Aggregate       Sort    Conjunct
```

```
    Leaf#01 BgrOnly PROD_DATEN Card=24063
      BgrNdx1 IDX_PROD_DATEN_SORT [1:1] Fan=8
  Inner loop      (zig-zag)
    Conjunct        Get     Retrieval by index of relation PRODUKTE
      Index name  IDX_PRODUKTE_SORT [0:0]
0 rows selected
```

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

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

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

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

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

This problem has been corrected in Oracle Rdb Release 7.0.6.1.

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

Bug 1581632

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

```
set flags 'strategy,detail';
select tt.employee_id, tt.last_name, jh.job_code
from
    (select e.employee_id, e.last_name
     from degrees d, employees e where
        e.employee_id = '00354'
        and d.employee_id = e.employee_id) as tt
    left outer join
    job_history jh
      on tt.last_name = '?' and                          <----
         jh.job_code = tt.employee_id;                   <----
Tables:
  0 = DEGREES
  1 = EMPLOYEES
  2 = JOB_HISTORY
Cross block of 2 entries        (Left Outer Join)
  Cross block entry 1
```

```
    Conjunct: "tt.last_name" = '?'
    Merge of 1 entries
      Merge block entry 1
      Cross block of 2 entries
        Cross block entry 1
          Get     Retrieval by index of relation 1:EMPLOYEES
            Index name  EMPLOYEES_HASH [1:1]       Direct lookup
              Key: 1.EMPLOYEE_ID = '00354'
        Cross block entry 2
          Index only retrieval of relation 0:DEGREES
            Index name  DEG_EMP_ID [1:1]
              Key: 0.EMPLOYEE_ID = 1.EMPLOYEE_ID
  Cross block entry 2
    Conjunct: ("tt.last_name" = '?') AND
              (2.JOB_CODE = tt.employee_id)
    Get     Retrieval by index of relation 2:JOB_HISTORY
      Index name  JH_EMPLOYEE_ID [0:0]
0 rows selected
```

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

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

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

There is no known workaround for this problem.

This problem has been corrected in Oracle Rdb Release 7.0.6.1.

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

Bug 1585429

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

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

```
        Index only retrieval of relation t2
          Index name  t2_ndx [0:0]
  Inner loop      (zig-zag)
    Aggregate-F1    Conjunct        Get
    Retrieval by index of relation t3
      Index name  t3_ndx [1:1]

          0
1 row selected
```

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

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

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

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

This problem has been corrected in Oracle Rdb Release 7.0.6.1.

5.1.16 Query Having Same SUBSTRINGs Within CASE Expression Returns Wrong Results

Bugs 1489972, 1485656, 975091

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

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

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

! Bug 975091
```

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

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

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

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

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

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

Another workaround is to use temporary variables for the substrings.

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

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

This problem has been corrected in Oracle Rdb Release 7.0.6.1.

5.1.17 AIJ File Name Was Not Translated When Defined in SQL

Bug 1573242

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

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

```
$define tmp1_aij tmp1:aij_01.aij

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

RMU/DUMP/HEADER my_db

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

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

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

RMU/DUMP/HEADER my_db

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

This problem has been corrected in Oracle Rdb Release 7.0.6.1.

5.1.18 Erroneous RDMS−F−ALSACTIVE Errors

Bug 1613851

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

This problem has been corrected in Oracle Rdb Release 7.0.6.1.

5.1.19 Aggregate Query With Nested MIN Function Returns Wrong Results

Bug 1408892

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

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

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

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

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

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

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

This problem has been corrected in Oracle Rdb Release 7.0.6.1.

5.2 SQL Errors Fixed

5.2.1 IMPORT of Multi−file Database as Single File Database May Fail

Bug 1365631

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

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

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

Note

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

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

5.2.2 Known Problems With EXPORT and IMPORT Fixed

Bug 1532755

In prior versions of Oracle Rdb 7.0, the EXPORT and IMPORT utilities did not correctly handle some Rdb metadata that was stored in LIST OF BYTE VARYING columns.

1. EXPORT did not correctly save the QUERY HEADER definition for domains and columns. EXPORT would introduce a CR/LF delimiter between each segment because it incorrectly treated the query header in the same way as a multi−line text string such as a comment or source definition. The use of this delimiter was introduced in Oracle Rdb7 to better handle large definitions and user comments.
   The workaround for this problem is to ALTER DOMAIN and ALTER TABLE ALTER COLUMN to apply a new and corrected QUERY HEADER to the affected domains and columns. Single segment query headers are not affected by this problem.
2. IMPORT did not correctly handle very long access control lists (ACL) for tables, databases, domains, functions or procedures. The ACL was truncated, usually with a corrupt final entry.
   The workaround for this problem is to use RMU/EXTRACT/ITEM=PROTECTION prior to the EXPORT and IMPORT so that the ACL can be reapplied to the database. In this case, a long ACL is one with more than 80 entries. In all likelihood, few databases would encounter this problem.
   The EXPORT interchange file (RBR) contains the correct information and so can be used successfully with this and later releases.

These problems have been corrected in Oracle Rdb Release 7.0.6.1.

5.2.3 Truncated Values Output by TRACE Statement

Bug 1231207

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

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

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

5.2.4 Multiple NOT NULL Constraints Generate WHYTWICE Exception

Bug 1293766

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

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

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

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

```
SQL>
SQL> show table tttt;
Information for table TTTT

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

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

This problem has been corrected in Oracle Rdb Release 7.0.6.1.

5.2.5 DROP FUNCTION or DROP PROCEDURE Leave Dependency Records

In prior releases of Oracle Rdb 7.0, the DROP FUNCTION and DROP PROCEDURE statements applied to stored routines did not correctly erase old dependency records from RDB$INTERRELATIONS. The result was that actions on these dependent objects would continue to fail, even though no relationship remained with the dropped routine.

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

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

```
SQL> create table sample1
cont>    (vector              integer(1),
cont>     pcnt                integer,
cont>     pv_ix               integer,
cont>     patt_ix             integer);
SQL>
SQL> create table sample2
cont>    (tchan               smallint,
cont>     chan_num            smallint,
cont>     mode                char(1),
cont>     pv_ix               integer);
SQL>
SQL> create procedure sethexbits
cont>     (inout char(544) by reference,
cont>      in smallint by value);
cont>     external name sethexbits
cont>         language C
cont>         general parameter style;
SQL>
SQL> create module vecsigmod language SQL
cont>   function vecsig(in :pv_ix integer) returns char(544);
cont>     begin
cont>       declare :hexvecstr char(544) = ' ';
cont>       call sethexbits(:hexvecstr, -1);
cont>       for :x
cont>         as each row of
```

```
cont>           select tchan from sample2 where pv_ix = :pv_ix
cont>       do
cont>         call sethexbits(:hexvecstr,:x.tchan);
cont>       end for;
cont>       return :hexvecstr;
cont>     end;
cont> end module;
SQL>
SQL> select cast(rdb$object_name as char(10)) as obj,
cont>        cast(rdb$subobject_name as char(10)) as subobj,
cont>        cast(rdb$entity_name1 as char(10)) as name1,
cont>        cast(rdb$entity_name2 as char(10)) as name2
cont> from rdb$interrelations
cont> where rdb$subobject_name = 'VECSIG'
cont>     or rdb$entity_name2 = 'VECSIG';
 OBJ          SUBOBJ       NAME1        NAME2
              SETHEXBITS   VECSIGMOD    VECSIG
 SAMPLE2                   VECSIGMOD    VECSIG
 SAMPLE2      PV_IX        VECSIGMOD    VECSIG
 SAMPLE2      TCHAN        VECSIGMOD    VECSIG
4 rows selected
SQL>
SQL> drop function vecsig;
SQL>
SQL> select cast(rdb$object_name as char(10)) as obj,
cont>        cast(rdb$subobject_name as char(10)) as subobj,
cont>        cast(rdb$entity_name1 as char(10)) as name1,
cont>        cast(rdb$entity_name2 as char(10)) as name2
cont> from rdb$interrelations
cont> where rdb$subobject_name = 'VECSIG'
cont>     or rdb$entity_name2 = 'VECSIG';
 OBJ          SUBOBJ       NAME1        NAME2
              SETHEXBITS   VECSIGMOD    VECSIG
 SAMPLE2                   VECSIGMOD    VECSIG
 SAMPLE2      PV_IX        VECSIGMOD    VECSIG
 SAMPLE2      TCHAN        VECSIGMOD    VECSIG
4 rows selected
SQL>
SQL> show function vecsig;
 No Functions Found
SQL>
SQL> show module vecsigmod;
 Module name is: VECSIGMOD
 Header:  vecsigmod language SQL
 No description found
 Owner is:
 Module ID is: 39
 No Procedures Found
 No Functions Found

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

This problem has been corrected in Oracle Rdb Release 7.0.6.1.

5.3 Oracle RMU Errors Fixed

5.3.1 RMU /UNLOAD /AFTER_JOURNAL Sort Performance

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

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

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

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

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

5.3.3 Confusing Lock Mode Displays Updated

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

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

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

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

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

and granted information. Locks on the waiting queue display only the requested mode; the granted mode is blank.

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

5.3.4 RMU/VERIFY Reports PGSPAMENT or PGSPMCLST Errors

Bug 1472061

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

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

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

   These problems have been corrected in Oracle Rdb Release 7.0.6.1. Further introduction of these SPAM inconsistencies should be reduced. Note that existing SPAM errors will remain until manually corrected. Also, while the incidence of these errors has been reduced they cannot be totally eliminated. See Section 9.0.6 for more information.

5.3.5 RMU/SHOW STATISTICS RMS−F−DEV Error With /INPUT

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

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

There is no known workaround for this problem.

This problem has been corrected in Oracle Rdb Release 7.0.6.1.

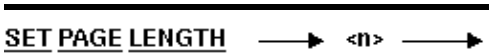
Chapter 6
Enhancements

6.1 Enhancements Provided in Oracle Rdb Release 7.0.6.4

6.1.1 New SET PAGE LENGTH Command for Interactive SQL

SQL now includes a SET PAGE LENGTH statement to size the page. Currently this is only used by the pagination control in the SQL HELP command.

FORMAT



USAGE NOTES

- The integer value must be a value between 10 and 32767.
- SET PAGE LENGTH can be used to effectively disable the paging performed by help by setting the length to a high value such as 32000.
- The page length is automatically set upon entry to interactive SQL and is based on the OpenVMS terminal setting for this session.
- The SHOW DISPLAY command can be used to view the currently defined page length.

This example uses the SET PAGE LENGTH command to change the pagination length of HELP.

```
SQL> set page length 40;
SQL> show display
Output of the query header is enabled
Output of the row counter is enabled
Output using edit strings is enabled
Page length is set to 40 lines
Line length is set to 80 bytes
Display NULL values using "NULL"
```

6.1.2 RMU /UNLOAD /AFTER_JOURNAL Wildcard Table Names

The RMU /UNLOAD /AFTER_JOURNAL command now supports wildcard processing of table names. The asterisk (*) and the percent sign (%) wildcard characters can be used in the table name specification to select all tables that satisfy the components you specify. The asterisk (*) matches zero or more characters and the percent sign (%) matches a single character.

Further, for table names that contain wild card characters, if the first character of the string is a pound sign (#), the wildcard specification is changed to a "not matching" comparison. This allows exclusion of tables based on a wildcard specification. Note that the pound sign (#) designation is only evaluated when the table name specification contains an asterisk (*) or percent sign (%).

For example, a table name specification of "*" indicates that all tables in the database are to be selected. A table name specification of "FOO%" indicates that all table names that are four characters long and that start with the string "FOO" (such as "FOOD" and "FOOT") are to be selected.

6.1.3 New Options for SET FLAGS Statement

This release of Oracle Rdb adds two new SET FLAGS keywords.

- WATCH_CALL
  This keyword traces the execution of queries, triggers and stored functions and procedures. The output includes the name of the trigger, function or procedure or "unnamed" for an anonymous query. In most cases, a query can be named using the OPTIMIZE AS clause. It also includes the value of CURRENT_USER during the execution of that routine. CURRENT_USER may be inherited from any module that uses the AUTHORIZATION clause.
  This flag can be disabled using NOWATCH_CALL, by using SET NOFLAGS, or using SET FLAGS 'NONE'.
- WATCH_OPEN
  This keyword traces all queries executed on the database. This may include SQL runtime system queries to lookup table names as well as queries executed by the application. The output includes the 32 digit hex identifier, the same as used by the CREATE OUTLINE statement. This value uniquely identifies the query being executed.
  If a query is a stored routine (function or procedure) then the notation "(stored)" is appended. If the query is named then it will be classified as "(query)", otherwise it will be designated as "(unnamed)". See the examples below for details.
  This flag can be disabled using NOWATCH_OPEN, by using SET NOFLAGS, or using SET FLAGS 'NONE'.

Usage Notes

- These keywords can also be used with the RDMS$SET_FLAGS logical name.
- The RDMS$DEBUG_FLAGS value "Xa" can also be used to enable WATCH_CALL.
- The RDMS$DEBUG_FLAGS value "Xo" can also be used to enable WATCH_OPEN.
- When using interactive or dynamic SQL, both WATCH_CALL and WATCH_OPEN will generate trace lines for the queries performed by the SQL runtime system against the Rdb system tables. There is no mechanism to disable the trace of such information.
- These flags cause queries and routines to be modified to output this information. This might add some extra CPU overhead to the application while this information is collected. Even when the flags are disabled, there exists some overhead that is not eliminated until the module or query is released, usually at DISCONNECT time.

Examples

Example 1: WATCH_CALL

This example shows the output of WATCH_CALL for an INSERT statement which causes an AFTER INSERT trigger (AFTER_INSERT) to be executed which calls a SQL function WRITE_TEXT to trace the input data. It then traces a query named using the OPTIMIZE AS clause.

```
SQL> insert into SAMPLE_T values ('Fred');
~Xa: routine "(unnamed)", user=SMITHI
~Xa: routine "AFTER_INSERT", user=SMITHI
~Xa: routine "WRITE_TEXT", user=SMITHI
~Xt: Fred
1 row inserted
SQL> select * from SAMPLE_T
cont>     optimize as LOOKUP_SAMPLE_T;
~Xa: routine "LOOKUP_SAMPLE_T", user=SMITHI
 NEW_NAME
 Fred
1 row selected
```

Example 2: WATCH_OPEN

This example shows the output of WATCH_OPEN for the same INSERT statement as seen in Example 1.

```
SQL> insert into SAMPLE_T values ('Fred');
~Xo: Start Request B667E51E3625026EB7FFF3F4D3A16DC3 (unnamed)
~Xo: Start Request A8568053FE5A1A0852A1BE83A884016F "AFTER_INSERT" (query)
~Xo: Start Request 08AE59062657299B4768F6C2DFB6928E "WRITE_TEXT" (stored)
~Xt: Fred
1 row inserted
SQL>
SQL> select * from SAMPLE_T
cont>     optimize as LOOKUP_SAMPLE_T;
~Xo: Start Request F6025FAB1DD36B0DE0E52F3A9641BC5F "LOOKUP_SAMPLE_T" (query)
 NEW_NAME
 Fred
 Fred
2 rows selected
```

6.1.4 Enhancements to RMU Extract

Bug 2130670

This release of Oracle Rdb improves the extraction of complex metadata items such as views and triggers.

- Improved extract of derived tables.
  In previous versions, Rdb would use derived column names based on position, for example F1, F2, etc. In this release, Rdb tries to promote the column names from the base table into the derived column name list. The result should be a more readable representation of the view or trigger definition.
  In the following example, the column name EMPLOYEE_ID is propagated through the derived table. In previous releases, this would be named using a generic label F1.

  ```
  create view SAMPLE_V
      (EMPLOYEE_ID,
       COUNTS) as
      select
          C1.EMPLOYEE_ID,
          C1.F2
      from
      (select C2.EMPLOYEE_ID,
          (select count(*) from SALARY_HISTORY C3
          where (C3.EMPLOYEE_ID = C2.EMPLOYEE_ID))
          from JOB_HISTORY C2) as C1 ( EMPLOYEE_ID, F2 )
      order by C1.F2 asc;
  ```
- Improved extract of IS NOT NULL clause.
  RMU now pushes the NOT into the expression so that it reads more naturally. For example, in previous versions A IS NOT NULL would be extracted as the equivalent expression NOT (A IS NULL) but now extracts (A IS NOT NULL).

6.2 Enhancements Provided in Oracle Rdb Release 7.0.6.3

6.2.1 Field Widths Wider on Row Cache Overview Display

On the "Row Cache Overview" display, the width of the "Searches" column has been increased from 9 to 10 characters to allow a display of up to 4294967295 (after this value, the 32−bit counter wraps back to zero). In addition, the width of the cache name column is tied to the screen width. If the screen is set to be wide enough (over 90 columns), the full cache name will be displayed; normally, only the first 24 characters of the name is displayed.

Additionally, the comparison used when sorting by value on the "Row Cache Overview" display has been modified to be unsigned (rather than signed). This prevents some cases of very large values being sorted in an incorrect order.

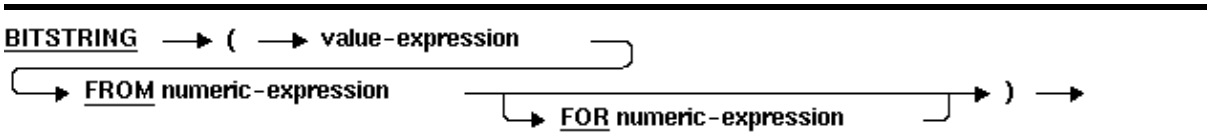
6.2.2 New BITSTRING Built In Function

Rdb now supports a BITSTRING function that can be used to extract selected bits from a binary data value. This functionality is primarily intended to query the bit values stored in the RDB$FLAGS columns in the Rdb system table but can also be used for user data.

BITSTRING accepts numeric and date/time values and processes them as bit arrays. The first (least significant) bit is numbered 1. The most significant bit depends on the data type.

- TINYINT has 8 bits
- SMALLINT has 16 bits
- INTEGER has 32 bits
- BIGINT, DATE, TIME, TIMESTAMP and INTERVAL types have 64 bits

FORMAT



USAGE NOTES

- The numeric expression after the FOR and FROM keywords must be an unscaled numeric value.
- If the numeric expression of the FOR clause is less than or equal to zero then it will be assumed equal to 1.
- If the FOR clause is omitted, it will default to a value that includes all remaining bits of the source value.
- If the FOR clause specifies a larger value than the number of bits remaining in the source then it will only return the remaining bits.

Example: Bit 1 in the RDB$FLAGS column of RDB$RELATIONS indicates that the table is a view. This example uses this query to fetch the names of all user defined views in the PERSONNEL database.

```
SQL> select rdb$relation_name
cont> from rdb$relations
cont> where rdb$system_flag = 0 and
```

```
cont>     bitstring (rdb$flags from 1 for 1) = 1;
 RDB$RELATION_NAME
 CURRENT_JOB
 CURRENT_SALARY
 CURRENT_INFO
3 rows selected
SQL>
```

6.2.3 RMU /SHOW STATISTICS Active User Stall Messages Sort by Process Id

The RMU /SHOW STATISTICS "Active User Stall Messages" display now includes the ability to sort the list of database users by process ID (OpenVMS PID). The Config option on the horizontal menu at the bottom of the screen can be used to control how the information is to be sorted. By default, the display is unsorted.

6.2.4 New Character Set ISOLATIN9

Bug 2063923

Oracle Rdb now supports the ISOLATIN9 character set (as described by ISO 8859−15).

ISOLATIN9 is similar to ISOLATIN1 except for 8 codepoints.

Table 6−1 ISOLATIN1 ISOLATIN9 Character Set Differences

	ISO Latin 1		ISO Latin 9	
Code Pos Hex	**Unicode Pos Hex**	**Name**	**Unicode Pos Hex**	**Name**
A4	00A4	currency symbol	20AC	euro sign
A6	00A6	broken bar	0160	latin capital letter s with caron
A8	00A8	diaeresis	0161	latin small letter s with caron
B4	00B4	acute accent	017D	latin capital letter z with caron
B8	00B8	cedilla	017E	latin small letter z with caron
BC	00BC	vulgar fraction one quarter	0152	latin capital ligature oe
BD	00BD	vulgar fraction one half	0153	latin small ligature oe
BE	00BE	vulgar fraction three quarters	0178	latin capital letter y with diaeresis

The main reason you might use ISOLATIN9 instead of ISOLATIN1 is due to the Euro character being correctly supported within ISOLATIN9.

6.2.5 Euro Character Now Supported Within the DEC_MCS Character Set

Bug 2063923

In line with changes made to the DEC MCS character set by Compaq to allow support of the Euro character, Oracle Rdb now supports the Euro character (hex A4) within the DEC_MCS character set.

6.2.6 RMU Support Added for New VMS Tape Density Values

Oracle Rdb RMU now supports the new VMS tape density and compression values introduced in VMS V7.2−1. The values that can be specified are the same values as those documented by VMS for the VMS INITIALIZE and MOUNT commands as well as other VMS commands that allow tape density and compression to be specified. The existing tape density values supported by the /DENSITY qualifier can continue to be specified for versions of VMS prior to VMS V7.2−1, for VMS tape device drivers that have not been enhanced to use these new density values, and even for VMS tape drivers that have been enhanced to use the new density values. However, if possible, the new density values should be specified for VMS tape device drivers that accept the new density values since in some cases, especially for newer tape drives and tape cartridges, the existing density values may not work as expected. This affects all RMU commands that support the /DENSITY qualifier: RMU /BACKUP, RMU /BACKUP /AFTER_JOURNAL and RMU /OPTIMIZE_AIJ. The new VMS tape density and compression values are sometimes referred to as "MTD" values (multiple tape density) or "MT3" (they translate to internal VMS values that start with "MT3$K_" while the existing density values translate to internal values that start with "MT$K_").

If the existing RMU tape density values are specified for VMS tape device drivers that support the new density values, they will be translated to the new density values if possible; otherwise a warning message will be issued and the existing tape density values will be used since the VMS tape driver that supports the new density values should accept the existing density values in most cases. Similarly, if the new tape density values are specified for VMS tape device drivers that do not support the new density values they will be translated to the existing density values if possible; otherwise a warning message will be issued and the new density value will be translated to the existing "DEFAULT" internal density value (MT$K_DEFAULT) since the tape device driver does not support the new density values. RMU queries the tape device driver at the start of the tape operation to determine if it supports the new density/compression values. If a density related error such as:

```
%RMU−E−DENSITY, TAPE_DEVICE:[000000]DATABASE.BCK; does not support specified
density
```

or

```
%RMU−E−POSITERR, error positioning TAPE_DEVICE:
```

or

```
%RMU−E−BADDENSITY, The specified tape density is invalid for this device
```

is returned, we recommend changing the value specified with the /DENSITY qualifier to one of the new density values for a VMS tape device driver that accepts the new density values or to one of the existing density values for a VMS tape device driver that accepts the existing density values. Generally, it is best to specify the new density values for tape device drivers that accept the new density values and the existing density values for tape device drivers that accept the existing density

values to be certain of achieving the desired tape density and compression. The warning message output if an existing density value cannot be translated to one of the new density values is:

```
%RMU_W_MTDSUPPORT, The specified density cannot be translated to an equivalent
 multiple tape density value
```

The warning message output if a new density value cannot be translated to one of the existing density values and is translated to the "DEFAULT" density value is:

```
%RMU-W-NOMTDSUPPORT, The specified multiple tape density cannot be translated
to an equivalent tape density value
```

The default behavior if the /DENSITY qualifier is not specified is to use the current tape density the tape has been set to by a VMS command such as MOUNT or INITIALIZE.

The existing syntax can continue to be used for the existing density values.

```
/DENSITY = density_value
```

where density_value can be one of the following numeric values:

```
0
1
2
800
833
1250
1600
6250
10000
10625
39782
39872
40000
70000
79564
79744
80000
160000
```

For the existing values, compression is determined by the density value and is not specified. For the value to be used for a particular tape drive and tape cartridge, we refer you to the VMS documentation.

For the new values, the syntax to be used is:

```
/DENSITY = new_density_value
```

where new_density_value can be one of the following values:

```
DEFAULT
800
833
1600
6250
3480
3490E
TK50
TK70
TK85
```

```
TK86
TK87
TK88
TK89
QIC
8200
8500
8900
DLT8000
SDLT
DDS1
DDS2
DDS3
DDS4
AIT1
AIT2
AIT3
AIT4
COMPACTION
NOCOMPACTION
```

If the new density values and the existing density values are the same (800,833,1600,6250), the intended value will be interpreted as a new value if the tape device driver accepts the new values and as an existing value if the tape device driver only accepts existing values.

For the new values which accept tape compression, the following syntax can be used:

```
/DENSITY = (new_density_value,[NO]COMPACTION)
```

To be used with the second "COMPACTION" parameter, the new density value must be one of the following new density values which accepts compression:

```
DEFAULT
3480
3490E
8200
8500
8900
TK87
TK88
TK89
DLT8000
SDLT
AIT1
AIT2
AIT3
AIT4
DDS1
DDS2
DDS3
DDS4
```

For the value to be used for a particular tape drive and cartridge, we refer you to the VMS documentation.

USAGE NOTES:

* If a density value is desired that is not supported by this syntax, use the VMS INITIALIZE and MOUNT commands to set the tape density and do not specify the /DENSITY qualifier.

* Please refer to the COMPAQ VMS documentation for detailed information on these density values and the tape drives and tape cartridges they should be used with.

* The same density syntax used on the command line can be specified in the PLAN file for PARALLEL RMU backup to tape.

The following example uses an existing density value.

```
$ RMU /BACKUP /DENSITY=1250 /REWIND /LABEL=(LABEL1,LABEL2)
    MF_PERSONNEL TAPE1:MFP.BCK, TAPE2:
```

The following example uses a new density value with no compression.

```
$ RMU /BACKUP /DENSITY=TK89 /REWIND /LABEL=(LABEL1,LABEL2)
    MF_PERSONNEL TAPE1:MFP.BCK, TAPE2:
```

The following example uses the same density value as above but calls for compression.

```
$ RMU /BACKUP /DENSITY=(TK89,COMPACTION) /REWIND /LABEL=(LABEL1,LABEL2)
    MF_PERSONNEL TAPE1:MFP.BCK, TAPE2:
```

6.3 Enhancements Provided in Oracle Rdb Release 7.0.6.2

6.3.1 Duplicate Node Algorithm Improved

In the case of an index with large index nodes (for example, larger than 5000 bytes), and small numbers of duplicates, the duplicate nodes created by CREATE INDEX would be much larger than needed. When the database had many of these duplicate nodes, the result could be a great waste of space in the database. An example of such an index would be an index on EMPLOYEE_ID in a table containing exactly two records per employee.

Prior to the current update, a workaround would be to use SORTED RANKED indices.

Earlier versions of Rdb would create duplicate nodes of length equal to the full node size, half the full node size, one−quarter, or one eighth of the full node size depending on the number of duplicates found while creating the index. Provision was thus allowed for adding duplicates as the database grew.

The current and future versions of Rdb recognize that even these reduced sizes may be too large in today's very large databases and continue halving to allow 1/16, 1/32, 1/64, or 1/128 of the full node size or, if those are still too large and there are ten or fewer duplicates, the smallest node size allocated is 80 bytes plus overhead (112). In any event, this latter size (112) is the smallest duplicate node allocated.

6.3.2 ALTER TABLE Support Extended for Temporary Tables

In prior releases of Oracle Rdb 7.0, temporary tables were not permitted to be altered using ALTER TABLE.

ALTER TABLE can now be used on temporary tables to add and drop constraints. The side effect of this change is that the IMPORT DATABASE command can now fully import a global or local temporary table that includes constraint definitions. In prior releases, an error such as this might have been generated.

```
SQL> import database
cont>     from saved_temps
cont>     filename new_temps;
   .
   .
   .
IMPORTing STORAGE AREA: RDB$SYSTEM
IMPORTing table EMPLOYEES
IMPORTing table EMPLOYEES_CLONE
%SQL−F−NOCONRES, unable to import constraint
%RDB−E−NO_META_UPDATE, metadata update failed
−RDMS−F−WISH_LIST, feature not implemented yet
%SQL−F−NOCONRES, unable to import constraint
%RDB−E−NO_META_UPDATE, metadata update failed
−RDMS−F−WISH_LIST, feature not implemented yet
%SQL−F−NOCONRES, unable to import constraint
%RDB−E−NO_META_UPDATE, metadata update failed
−RDMS−F−WISH_LIST, feature not implemented yet
SQL>
```

```
SQL> show table (constraint) EMPLOYEES_CLONE
Information for table EMPLOYEES_CLONE

Temporary Global
On commit Preserve rows

Table constraints for EMPLOYEES_CLONE:
No constraints found

Constraints referencing table EMPLOYEES_CLONE:
No constraints found

SQL> disconnect all;
```

Alter commands which change the record layout are still not allowed. These include adding and dropping columns and modifying the data type of a column. However, most other ALTER TABLE operations are now supported for local and global temporary tables.

This change has been made in Oracle Rdb Release 7.0.6.2.

6.3.3 New Minimum Value for the INTERVAL Leading Precision

In prior releases of Oracle Rdb, the minimum value for the interval leading precision was restricted to two digits. This restriction has been removed. An interval leading precision of 1 is now supported.

The following example shows the support for the lower precision value.

```
SQL> create table TIME_CLOCK
cont>     (employee_id  char(5),
cont>      clock_on             timestamp (2),
cont>      clock_off            timestamp (2),
cont>      shift_duration
cont>          computed by (clock_off - clock_on) hour (1) to minute);
SQL>
SQL> show table (column) TIME_CLOCK
Information for table TIME_CLOCK

Columns for table TIME_CLOCK:
Column Name                    Data Type        Domain
-----------                    ---------        ------
EMPLOYEE_ID                    CHAR(5)
CLOCK_ON                       TIMESTAMP(2)
CLOCK_OFF                      TIMESTAMP(2)
SHIFT_DURATION                 INTERVAL
                                HOUR (1) TO MINUTE
 Computed:       by (clock_off - clock_on) hour (1) to minute
```

As in previous releases, if no precision is provided then a default of 2 digits will be used.

This change has been made in Oracle Rdb Release 7.0.6.2.

6.3.4 Command Line Recall Expanded to 255 Lines

In prior releases of Oracle Rdb, the command line recall was limited to the last 20 lines. This limit has been lifted to 255 (the maximum supported by OpenVMS) for this release of Rdb.

If more recall is required, SQL provides the EDIT command to edit whole statements. This interface currently saves the last 20 commands for edit, but the SET EDIT KEEP statement can be used to expand this number.

This change has been made in Oracle Rdb Release 7.0.6.2.

6.3.5 RMU /REPAIR /INITIALIZE ONLY_LAREA_TYPE Keyword

A new ONLY_LAREA_TYPE keyword has been added to the RMU /REPAIR /INITIALIZE qualifier. This keyword, along with the /NOSPAM and /NOABM qualifiers, allows only the logical area "type" field to be updated in the AIP (area inventory pages). Avoiding SPAM page updates significantly improves performance of this operation.

The RMU /UNLOAD /AFTER_JOURNAL and RMU /SHOW STATISTICS commands use the on−disk area inventory pages (AIPs) to determine the appropriate "type" of each logical area. However, this logical area information in the AIP is generally unknown for logical areas created prior to Oracle Rdb V7.0.1. If the RMU /UNLOAD /AFTER_JOURNAL command cannot determine the logical area type for one or more AIP entries, a warning message is displayed for each such area and may ultimately return logical DBKEYs with a "0" (zero) area number for records stored in mixed format storage areas.

In order to update the on disk logical area "type" in the AIP, the RMU /REPAIR utility must be used. The /INITIALIZE = LAREA_PARAMETERS =optionfile qualifier option file can be used with the /TYPE qualifier. For example, to repair the EMPLOYEES table of the MF_PERSONNEL database, you would create an options file that contains the following line:

```
EMPLOYEES /TYPE=TABLE
```

For partitioned logical areas, the /AREA=name qualifier can be used to identify the specific storage areas that are to be updated. For example, to repair the EMPLOYEES table of the MF_PERSONNEL database for the EMPID_OVER storage area only, you would create an options file that contains the following line:

```
EMPLOYEES /AREA=EMPID_OVER /TYPE=TABLE
```

The /TYPE qualifier specifies the type of a logical area. The following keywords are allowed:

- TABLE − Specifies that the logical area is a data table. This would be a table created using the SQL "CREATE TABLE" syntax.
- BTREE − Specifies that the logical area is a b−tree index. This would be an index created using the SQL "CREATE INDEX TYPE IS SORTED" syntax.
- HASH − Specifies that the logical area is a hash index. This would be an index created using the SQL "CREATE INDEX TYPE IS HASHED" syntax.
- SYSTEM − Specifies that the logical area is a system record which is used to identify hash buckets. Users cannot explicitly create these types of logical areas. This type should NOT be used for the RDB$SYSTEM logical areas. This type does NOT identify system relations.
- BLOB − Specifies that the logical area is a blob (segmented string; list of byte varying) repository.

There is no explicit error checking of the "type" specified for a logical area. However, an incorrect type may cause the RMU /UNLOAD /AFTER_JOURNAL command to be unable to correctly return valid logical DBKEYs.

The ONLY_LAREA_TYPE keyword can be specified along with the /NOSPAM and /NOABM qualifiers to cause *only* the logical area type field to be updated in the area inventory pages. All other actions specified in the options file are ignored when ONLY_LAREA_TYPE is specified. By updating only the logical area type in the AIP entries and not the SPAM pages, the RMU /REPAIR operation can be considerably faster.

6.3.6 New /TRANSACTION_TYPE Qualifier for RMU /Unload

This qualifier provides complete read transaction control to the user.

SYNTAX

/TRANSACTION_TYPE=options

One of the following transaction modes can be specified:

```
AUTOMATIC
READ_ONLY
EXCLUSIVE
PROTECTED
SHARED
```

- AUTOMATIC
  The transaction type will depend upon the current database settings for snapshots (enabled, deferred, or disabled), transaction modes available to this user, and the standby status of this database. AUTOMATIC is the default transaction mode.
- READ_ONLY
  Starts a READ ONLY transaction.
- EXCLUSIVE
  Starts a READ WRITE transaction and reserves the table for EXCLUSIVE READ.
- PROTECTED
  Starts a READ WRITE transaction and reserves the table for PROTECTED READ.
- SHARED
  Starts a READ WRITE transaction and reserves the table for SHARED READ.

The transaction isolation level can be specified using the ISOLATION_LEVEL option. It accepts the following keywords:

```
READ_COMMITTED
REPEATABLE_READ
SERIALIZABLE
```

Please refer to the Oracle Rdb7 SQL Reference Manual under the SET TRANSACTION statement for a complete description of these isolation levels.

The wait setting can be specified using:

```
WAIT [ = n ]
NOWAIT
```

Wait accepts an optional integer value representing the number of seconds to wait before the transaction times out.

- WAIT
  Will wait indefinitely on a locked resource.
- WAIT = n

'n' is the transaction lock timeout interval. This instructs Rdb to wait 'n' seconds before aborting the wait, and the RMU Unload session. Specifying a wait timeout interval of zero (0) is equivalent to specifying NOWAIT.

- NOWAIT
  Will not wait on locked resources.

USAGE NOTES

* If the /TRANSACTION_TYPE qualifier is omitted, then a READ ONLY transaction is started against the database. This is provided for backward compatibility with prior Rdb releases. However, if the /TRANSACTION_TYPE qualifier is used without specifying a transaction mode then AUTOMATIC will be used.

* If the database has snapshots disabled, then Oracle Rdb will default to a READ WRITE ISOLATION LEVEL SERIALIZABLE transaction. Locking may be reduced by specifying AUTOMATIC, or (SHARED,ISOLATION_LEVEL=READ_COMMITTED) transaction.

EXAMPLES

Example 1: Specify a transaction for RMU /UNLOAD equivalent to the SET TRANSACTION READ WRITE WAIT 36 RESERVING table1 FOR SHARED WRITE;

```
$ RMU /UNLOAD-
    /TRANSACTION_TYPE=(SHARED,ISOLATION=REPEAT,WAIT=36)-
    SAMPLE.RDB-
    TABLE1-
    TABLE.DAT
```

Example 2: This example specifies the options which were the default transaction style in prior releases.

```
$ RMU /UNLOAD-
    /TRANSACTION_TYPE=(READ_ONLY,ISOLATION_LEVEL=SERIALIZABLE)-
    SAMPLE.RDB-
    TABLE1-
    TABLE1.DAT
```

Example 3: This example specifies transaction type of AUTOMATIC so that RMU/ Unload will adapt to the current database configuration. For instance, if the database currently has SNAPSHOTS ENABLED DEFERRED, it is more efficient to start a READ WRITE transaction with isolation level READ COMMITTED. This allows the transaction to start immediately (a READ ONLY transaction may stall), and the selected isolation level keeps row locking to a minimum.

```
$ RMU/UNLOAD-
    /TRANSACTION_TYPE=(AUTOMATIC)-
    SAMPLE.RDB-
    TABLE1-
    TABLE1.DAT
```

This could also be explicitly stated using:

```
$ RMU/UNLOAD-
    /TRANSACTION_TYPE=(SHARED,ISOLATION=READ_COMMITTED)-
    SAMPLE.RDB-
    TABLE1-
    TABLE1.DAT
```

6.3.7 New /TRANSACTION_TYPE Qualifier for RMU /EXTRACT

This qualifier provides complete read transaction control to the user.

SYNTAX

/TRANSACTION_TYPE=options

One of the following transaction modes can be specified:

```
AUTOMATIC
READ_ONLY
WRITE
```

- AUTOMATIC
  The transaction type will depend upon the current database settings for snaphots (enabled, deferred, or disabled), transaction modes available to this user, and the standby status of this database.
- READ_ONLY
  Starts a READ ONLY transaction.
- WRITE
  Starts a READ WRITE transaction.

The transaction isolation level can be specified using the ISOLATION_LEVEL option. It accepts the following keywords:

```
READ_COMMITTED
REPEATABLE_READ
SERIALIZABLE
```

Please refer to the Oracle Rdb7 SQL Reference Manual under the SET TRANSACTION statement for a complete description of these isolation levels.

The wait setting can be specified using:

```
WAIT [ = n ]
NOWAIT
```

Wait accepts an optional integer value representing the number of seconds to wait before the transaction times out.

- WAIT
  Will wait indefinitely on a locked resource.
- WAIT = n
  'n' is the transaction lock timeout interval. This instructs Rdb to wait 'n' seconds before aborting the wait, and the RMU /EXTRACT session. Specifying a wait timeout interval of zero (0) is equivalent to specifying NOWAIT.
- NOWAIT
  Will not wait on locked resources.

USAGE NOTES

* If the qualifier /TRANSACTION_TYPE is omitted from the command line then the default transaction will be READ ONLY, which is backward compatible with prior releases of Oracle Rdb. If /TRANSACTION_TYPE is specified with no options then the default is

/TRANSACTION_TYPE=AUTOMATIC.

Note: If RMU /EXTRACT detects that the database has snapshots disabled and /TRANSACTION_TYPE was omitted, then the transaction is restarted as READ WRITE ISOLATION LEVEL READ COMMITTED to reduce the number of rows locked by operations /OPTION=VOLUME_SCAN.

* Although a WRITE transaction is started on the database, RMU Extract does not attempt to write to the database tables.

EXAMPLES

Example 1: If a database has SNAPSHOTS ENABLED DEFERRED it may be preferable to start a READ WRITE transaction. In this environment, using READ ONLY will cause a switch to a temporary SNAPSHOTS ENABLED IMMEDIATE state. This transition will force the READ ONLY transaction to wait while all READ WRITE transactions complete, and then all new READ WRITE transactions performing updates will start writing rows to the snapshot files for use by possible read only transactions. To avoid this problem, use an RMU /EXTRACT command specifying a READ WRITE ISOLATION LEVEL READ COMMITTED transaction.

```
$ RMU/EXTRACT/ITEM=TABLE/OUT=TABLES.SQL-
    /TRANSACTION_TYPE=(WRITE,ISOLATION=READ)-
    SAMPLE.RDB
```

Example 2: This example specifies the options which were the default transaction style in prior releases.

```
$ RMU/EXTRACT/ITEM=TABLE/OUT=TABLES.SQL-
    /TRANSACTION_TYPE=(READ_ONLY)-
    SAMPLE.RDB
```

Example 3: This example specifies transaction type of AUTOMATIC so that RMU/ Extract will adapt to the current database configuration. For instance, if the database currently has SNAPSHOTS ENABLED DEFERRED, it is more efficient to start a READ WRITE transaction with isolation level READ COMMITTED. This allows the transaction to start immediately (a READ ONLY transaction may stall), and the selected isolation level keeps row locking to a minimum.

```
$ RMU/EXTRACT-
    /TRANSACTION_TYPE=(AUTOMATIC)-
    SAMPLE.RDB
```

This could also be explicitly stated using:

```
$ RMU/EXTRACT-
    /TRANSACTION_TYPE=(WRITE,ISOLATION=READ_COMMITTED)-
    SAMPLE.RDB
```

6.3.8 Installing Oracle Rdb Images as Resident on OpenVMS Alpha

On OpenVMS Alpha systems, performance of applications using Oracle Rdb may improve by installing several of the Oracle Rdb images as "resident" with the OpenVMS Install utility (INSTALL). Installing images as resident allows them to take advantage of the OpenVMS Alpha image−slicing features.

The code sections of an image installed as resident reside in huge pages called granularity hint regions (GHRs) in memory. The OpenVMS Alpha hardware can consider a set of pages as a single GHR. This GHR can be mapped by a single page table entry (PTE) in the translation buffer (TB). The result is a reduction in TB miss rates. For more information on slicing shareable images, see the OpenVMS documentation set.

Further, OpenVMS versions starting with V7.2−1H1 support "Resource Affinity Domains" or RADs. When RAD support is enabled, VMS can replicate /RESIDENT installed image data on each RAD. The advantage to this replication is that any CPU access to the image memory will always be in the same RAD.

In order to take advantage of this capability, the image must be installed in the system startup procedure before the end of SYSTARTUP_VMS.COM. The easiest way to accomplish this for the Oracle Rdb images is to execute SYS$STARTUP:RMONSTART70.COM from SYSTARTUP_VMS.COM (the site−specific system startup procedure).

If you use many resident images, you may need to modify the GH_RES_CODE system parameter to add approximately 2048 additional pages. The System Dump Analyzer (SDA) command "CLUE MEMORY/GH/FULL" can be used to display the contents and free space within the "Resident Image Code Region".

To install images as resident, use a text editor to modify the the command procedures RMONSTART70.COM and SQL$STARTUP.COM located in the SYS$STARTUP directory. Remove the comment character (!) from the line RESIDENT = "/RESIDENT" and then several Rdb shareable images will be installed as /RESIDENT.

6.3.9 New DUMP Output Format for LogMiner

A new output format type of "DUMP" has been added to the RMU /UNLOAD /AFTER_JOURNAL command. This output format is intended solely as a debug and informational tool. For each column of a record, the first 200 bytes of data contents are formatted such that binary numeric fields are converted to text and text fields are displayed with periods (.) replacing non−printable characters. NULL columns are indicated with the character string "NULL". The actual data length is indicated for VARCHAR columns.

Example output with the /FORMAT=DUMP qualifier:

```
$ RMU /UNLOAD /AFTER_JOURNAL MYDB.RDB MYDB.AIJBCK /FORMAT=DUMP
    /TABLE=(NAME=ALL_DATATYPES_TBL, OUTPUT=SYS$OUTPUT:)
RDB$LM_ACTION                   : M
RDB$LM_RELATION_NAME            : ALL_DATATYPES_TBL
RDB$LM_RECORD_TYPE              : 25
RDB$LM_DATA_LEN                 : 460
RDB$LM_NBV_LEN                  : 66
RDB$LM_DBK                      : 46:635:0
RDB$LM_START_TAD                : 21−JUL−2001 15:48:52.6512009
RDB$LM_COMMIT_TAD               : 21−JUL−2001 15:48:53.0586846
RDB$LM_TSN                      : 160
RDB$LM_REC_VER                  : 1
TINT                            : −123
SINT                            : −321
INTEGER                         : −212
BINT                            : NULL
DECIMAL                         : −145
NUMERIC                         : NULL
FLOAT                           : −1.000000000000000E+000
DOUBLE_PRECISION                : −2.000000000000000E+000
```

```
CHAR1                           : A
CHAR20                          : ABCDEFGHIJKLMNOPQRST
VCHAR_COL                       : (10) ABCDEFGHIJ
```

Note

The contents and format of the output when the /FORMAT=DUMP qualifier is specified may change in the future.

If needed, the record definition (.RRD) file may be used to determine the actual data type for each field of the table(s) being extracted.

6.3.10 Data and Spam Prefetch Screens Added to RMU /SHOW Statistics

Two new screens have been added to the PIO statistics part of RMU /SHOW statistics. These screens display prefetch statistics (APF and DAPF). In prior versions, the DAPF statistics were displayed on the "Fetch" screens. Those statistics were moved to the new prefetch screens. In addition, APF statistics are now displayed on the new screens as well. An example is provided below.

```
Node: NODE1 (1/1/1)     Oracle Rdb V7.0-62 Perf. Monitor  6-AUG-2001 10:28:10.65
Rate: 3.00 Seconds      PIO Statistics--Data Prefetches     Elapsed: 00:58:17.86
Page: 1 of 1                    DEV:[DIR]DB.RDB                     Mode: Online
------------------------------------------------------------------------------

statistic.........      rate.per.second.............. total....... average......
name..............      max..... cur..... avg....... count....... per.trans....

APF start:success              0       0       0.4          872           1.0
         :failure              0       0       0.0          101           0.1

APF I/O: utilized              0       0       0.4          872           1.0
       : wasted                0       0       0.0            0           0.0


DAPF start:success             0       0       0.0           74           0.0
          :failure             0       0       0.0           62           0.0

DAPF I/O: utilized             0       0       0.0           18           0.0
        : wasted               0       0       0.0           56           0.0
```

The information on these screens may be used to determine the effectiveness of the APF and DAPF features. The individual rows may be interpreted as follows:

- The "APF start:success" statistics shows how many times Oracle Rdb successfully initiated an I/O to prefetch a buffer.
- The "APF start:failure" statistics shows how many times Oracle Rdb attempted to initiate a prefetch but was unable to obtain the necessary buffer lock to proceed.
- The "APF I/O: utilized" statistics shows how many times Oracle Rdb actually used a buffer that was prefetched.
- The "APF I/O: wasted" statistics shows how many times Oracle Rdb prefetched a buffer but never actually used it.

6.4 Enhancements Provided in Oracle Rdb Release 7.0.6.1

6.4.1 RMU /UNLOAD /AFTER_JOURNAL Database Metadata File

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

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

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

- SAVE_METADATA=filename
  This qualifier indicates that the RMU /UNLOAD /AFTER_JOURNAL command is to write metadata information to the specified file. The RESTORE_METADATA, TABLE and OPTIONS=FILE qualifiers and the AIJ_NAME parameter are not allowed when the SAVE_METADATA qualifier is present. The default file type is ".METADATA".
- RESTORE_METADATA=filename
  This qualifier indicates that the RMU /UNLOAD /AFTER_JOURNAL command is to read database metadata information from the specified file. The DATABASE parameter is required but the database itself is not accessed when the RESTORE_METADATA qualifier is specified. The default file type is ".METADATA".

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

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

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

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

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

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

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

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

Chapter 7
Oracle Rdb Continuous LogMiner

Oracle Rdb Continuous LogMiner™ (sometimes referred to as CLM) is an extension to the existing Oracle Rdb LogMiner feature. It allows online extraction of after−image journal data in "near−real" time. The Continuous LogMiner functionality extends Oracle Rdb LogMiner operations from off−line backup .aij files to live online AIJ information.

This chapter describes the Oracle Rdb LogMiner feature and the changes and enhancements made to Oracle Rdb to support Continuous LogMiner functionality. It includes all of the LogMiner documentation that the next release of the *Oracle RMU Reference Manual* will contain.

7.1 RMU Unload After_Journal Command

Format

RMU/Unload/After_Journal root-file-spec aij-file-name

Command Qualifiers	Defaults
/Before=date-time	None
/Continuous	/NoContinuous
/Extend_Size=integer	/Extend_Size=1000
/Format=options	See description
/Include=Action=(include-type)	Include=Action=(NoCommit,Modify,Delete)
/IO_Buffers=integer	/IO_Buffers=2
/[No]Log	Current DCL verify value
/Options=options-list	See description
/Order_AIJ_files	/NoOrder_aij_files
/Output=file-spec	/Output=SYS$OUTPUT
/Parameter=character-strings	None
/Restart=(restart-point)	None
/Restore_Metadata=file-spec	None
/Save_Metadata=file-spec	None
/Select=selection-type	/Select=Commit_Transaction
/Since=date-time	None
/Sort_Workfiles=integer	/Sort_Workfiles=2
/Statistics_Interval=integer	See description
/Table=(Name=table-name, [table-options ...])	See description
[No]Trace	None
	/Notrace

DESCRIPTION

The RMU Unload After_Journal command translates the binary data record contents of an after−image journal (.aij) file into an output file. Data records for the specified tables for committed transactions are extracted to an output stream (file, device, or application callback) in the order that the transactions were committed.

Before you use the RMU Unload After_Journal command, you must enable the database for LogMiner extraction. Use the RMU Set Logminer command to enable the LogMiner for Rdb feature for the database. Before you use the RMU Unload After_Journal command with the Continuous qualifier, you must enable the database for Continuous LogMiner extraction. See Section 7.2 for more information.

Data records extracted from the .aij file are those records that transactions added, modified, or deleted in base database tables. Index nodes, database metadata, segmented strings (BLOB), views, COMPUTED BY columns, system relations, and temporary tables cannot be unloaded from after−image journal files.

For each transaction, only the final content of a record is extracted. Multiple changes to a single record within a transaction are condensed so that only the last revision of the record appears in the output stream. You cannot determine which columns were changed in a data record directly from the after−image journal file. In order to determine which columns were changed, you must compare the record in the after−image journal file with a previous record.

The database used to create the after−image journal files being extracted must be available during the RMU Unload After_Journal command execution. The database is used to obtain metadata information (such as table names, column counts, record version, and record compression) needed to extract data records from the .aij file. The database is read solely to load the metadata and is then detached. Database metadata information can also be saved and used in a later session. See the Save_MetaData and Restore_MetaData qualifiers for more information.

If you use the Continuous qualifier, the database must be opened on the node where the Continuous LogMiner process is running. The database is always used and must be available for both metadata information and for access to the online after−image journal files. The Save_MetaData and Restore_MetaData qualifiers are not permitted with the Continuous qualifier.

When one or more .aij files and the Continuous qualifier are both specified on the RMU Unload After_Journal command line, it is important that no .aij backup operations occur until the Continuous LogMiner process has transitioned to online mode (where the active online .aij files are being extracted). If you are using automatic .aij backups and wish to use the Continuous LogMiner feature, Oracle recommends that you consider disabling the automatic backup feature (ABS) and use manual .aij backups so that you can explicitly control when .aij backup operations occur.

The after−image journal file or files are processed sequentially. All specified tables are extracted in one pass through the after−image journal file.

As each transaction commit record is processed, all modified and deleted records for the specified tables are sorted to remove duplicates. The modified and deleted records are then written to the output streams. Transactions that were rolled back are ignored. Data records for tables that are not being extracted are ignored. The actual order of output records within a transaction is not predictable.

In the extracted output, records that were modified or added are returned as being modified. It is not possible to distinguish between inserted and updated records in the output stream. Deleted (erased) records are returned as being deleted. A transaction that modifies and deletes a record generates only a deleted record. A transaction that adds a new record to the database and then deletes it within the same transaction generates only a deleted record.

The LogMiner process signals that a row has been deleted by placing a D in the RDB$LM_ACTION field. The contents of the row at the instant before the delete operation are recorded in the user fields of the output record. If a row was modified several times within a transaction before being deleted, the output record contains only the delete indicator and the results of the last modify operation. If a row is inserted and deleted in the same transaction, only the delete record appears in the output.

Records from multiple tables can be output to the same or to different destination streams. Possible output destination streams include the following:

- ◊ File
- ◊ OpenVMS Mailbox
- ◊ OpenVMS Pipe

◊ Direct callback to an application through a run−time activated shareable image

COMMAND PARAMETERS

root−file−spec

The root file specification of the database for the after−image journal file from which tables will be unloaded. The default file extension is .rdb.

The database must be the same actual database that was used to create the after−image journal files. The database is required so that the table metadata (information about data) is available to the RMU Unload After_Journal command. In particular, the names and relation identification of valid tables within the database are required along with the number of columns in the table and the compression information for the table in various storage areas.

The RMU Unload After_Journal process attaches to the database briefly at the beginning of the extraction operation in order to read the metadata. Once the metadata has been read, the process disconnects from the database for the remainder of the operation unless the Continuous qualifier is specified. The Continuous qualifier indicates that the extraction operation is to run non−stop, and the process remains attached to the database.

aij−file−name

One or more input after−image journal backup files to be used as the source of the extraction operation. Multiple journal files can be extracted by specifying a comma−separated list of file specifications. Oracle RMU supports OpenVMS wildcard specifications (using the * and % characters) to extract a group of files. A file specification beginning with the at (@) character refers to an options file containing a list of after−image journal files (rather than the file specification of an after−image journal itself). If you use the at character syntax, you must enclose the at character and the file name in double quotation marks (for example, specify aij−file−name as "@files.opt"). The default file extension is .aij.

COMMAND QUALIFIERS

Before=date−time

Specifies the ending time and date for transactions to be extracted. Based on the Select qualifier, transactions that committed or started prior to the specified Before date are selected. Information changed due to transactions that committed or started after the Before date is not included in the output.

Continuous

NoContinuous

Causes the LogMiner process to attach to the database and begin extracting records in "near−real" time. When the Continuous qualifier is specified, the RMU Unload After_Journal command extracts records from the online after−image journal files of

the database until it is stopped via an external source (for example, Ctrl/y, STOP/ID, $FORCEX, or database shutdown).

A database must be explicitly enabled for the Continuous LogMiner feature. To enable the Continuous LogMiner feature, use the RMU Set Logminer command with the Enable and Continuous qualifiers; to disable use of the Continuous LogMiner feature, use the RMU Set Logminer command with the Enable and Nocontinuous qualifiers.

The output from the Continuous LogMiner process is a continuous stream of information. The intended use of the Continuous LogMiner feature is to write the changes into an OpenVMS mailbox or pipe, or to call a user−supplied callback routine. Writing output to a disk file is completely functional with the Continuous LogMiner feature, however, no built−in functionality exists to prevent the files from growing indefinitely.

It is important that the callback routine or processing of the mailbox be very responsive. If the user−supplied callback routine blocks, or if the mailbox is not being read fast enough and fills, the RMU Unload After_Journal command will stall. The Continuous LogMiner process prevents backing up the after−image journal that it is currently extracting along with all subsequent journals. If the Continuous LogMiner process is blocked from executing for long enough, it is possible that all available journals will fill and will not be backed up.

When a database is enabled for the Continuous LogMiner feature, an AIJ "High Water" lock (AIJHWM) is utilized to help coordinate and maintain the current .aij end−of−file location. The lock value block for the AIJHWM lock contains the location of the highest written .aij block. The RMU Unload After_Journal command with the Continuous qualifier polls the AIJHWM lock to determine if data has been written to the .aij file and to find the highest written block. If a database is not enabled for the Continuous LogMiner feature, there is no change in locking behavior; the AIJHWM lock is not maintained and thus the Continuous qualifier of the RMU Unload After_Journal command is not allowed.

In order to maintain the .aij end−of−file location lock, processes that write to the after−image journal file must use the lock to serialize writing to the journal. When the Continuous LogMiner feature is not enabled, processes instead coordinate allocating space in the after−image journal file and can write to the file without holding a lock. The Continuous LogMiner process requires that the AIJHWM lock be held during the .aij I/O operation. In some cases, this can reduce overall throughput to the .aij file as it serves to reduce multiple over−lapped I/O write operations by multiple processes.

The Save_Metadata and Restore_Metadata qualifiers are incompatible with the Continuous qualifier.

Extend_Size=integer

Specifies the file allocation and extension quantity for output data files. The default extension size is 1000 blocks. Using a larger value can help reduce output file fragmentation and can improve performance when large amounts of data are extracted.

Format=options

If the Format qualifier is not specified, Oracle RMU outputs data to a fixed−length binary flat file.

The format options are:

◊ Format=Binary
If you specify the Format=Binary option, Oracle RMU does not perform any data conversion; data is output in a flat file format with all data in the original binary state.
Table 7−1 describes the output fields and data types of an output record in Binary format.

Table 7−1 Output Fields

Field Name	Data Type	Byte Length	Description
ACTION	CHAR (1)	1	Indicates record state. "M" indicates an insert or modify action. "D" indicates a delete action. "E" indicates stream end−of−file (EOF) when a callback routine is being used. "P" indicates a value from the command line Parameter qualifier when a callback routine is being used (see Parameter qualifier). "C" indicates transaction commit information when the Include=Action=Commit qualifier is specified.
RELATION_NAME	CHAR (31)	31	Table name. Space padded to 31 characters.
RECORD_TYPE	INTEGER (LONGWORD)	4	The Oracle Rdb internal relation identifier.
DATA_LEN	SMALLINT (WORD)	2	Length, in bytes, of the data record content.
NBV_LEN	SMALLINT (WORD)	2	Length, in bits, of the null bit vector content.
DBK	BIGINT (QUADWORD)	8	Records logical database key. The database key is a 3−field structure containing a 16−bit line number, a 32−bit page number and a 16−bit area number.

START_TAD	DATE VMS (QUADWORD)	8	Date/time of the start of the transaction.
COMMIT_TAD	DATE VMS (QUADWORD)	8	Date/time of the commitment of the transaction.
TSN	BIGINT (QUADWORD)	8	Transaction sequence number of the transaction that performed the record operation.
RECORD_VERSION	SMALLINT (WORD)	2	Record version.
Record Data	Varies		Actual data record field contents.
Record NBV	BIT VECTOR (array of bits)		Null bit vector. There is one bit for each field in the data record. If a bit value is 1, the corresponding field is NULL; if a bit value is 0, the corresponding field is not NULL and contains an actual data value. The null bit vector begins on a byte boundary. Any extra bits in the final byte of the vector after the final null bit are unused.

◊ Format=Dump

If you specify the Format=Dump option, Oracle RMU produces an output format suitable for viewing. Each line of Dump format output contains the column name (including LogMiner prefix columns) and up to 200 bytes of the column data. Unprintable characters are replaced with periods (.), and numbers and dates are converted to text. NULL columns are indicated with the string "NULL". This format is intended to assist in debugging; the actual output contents and formatting will change in the future.

◊ Format=Text

If you specify the Format=Text option, Oracle RMU converts all data to printable text in fixed−length columns before unloading it. VARCHAR(n) strings are padded with blanks when the specified string has fewer characters than *n* so that the resulting string is *n* characters long.

◊ Format=(Delimited_Text [,delimiter−options])

If you specify the Format=Delimited_Text option, Oracle RMU applies delimiters to all data before unloading it.

DATE VMS dates are output in the collatable time format, which is yyyymmddhhmmsscc. For example, March 20, 1993 is output as: 1993032000000000.

Delimiter options are:

· Prefix=string

Specifies a prefix string that begins any column value in the ASCII output file. If you omit this option, the column prefix is a quotation mark (").

· Separator=string

Specifies a string that separates column values of a row. If you omit

this option, the column separator is a single comma (,).

· Suffix=string
Specifies a suffix string that ends any column value in the ASCII output file. If you omit this option, the column suffix is a quotation mark (").

· Terminator=string
Specifies the row terminator that completes all the column values corresponding to a row. If you omit this option, the row terminator is the end of the line.

· Null=string
Specifies a string that, when found in the database column, is unloaded as "NULL" in the output file.
The Null option can be specified on the command line as any one of the following:

- A quoted string
- An empty set of double quotes ("")
- No string

The string that represents the null character must be quoted on the Oracle RMU command line. You cannot specify a blank space or spaces as the null character. You cannot use the same character for the Null value and other Delimited_Text options.

Note

The values for each of the strings specified in the delimiter options must be enclosed within quotation marks. Oracle RMU strips these quotation marks while interpreting the values. If you want to specify a quotation mark (") as a delimiter, specify a string of four quotation marks. Oracle RMU interprets four quotation marks as your request to use one quotation mark as a delimiter. For example, Suffix= """".

Oracle RMU reads these quotation marks as follows:

- ***The first quotation mark is stripped from the string.***
- ***The second and third quotation mark are interpreted as your request for one quotation mark (") as a delimiter.***
- ***The fourth quotation mark is stripped.***

This results in one quotation mark being used as a delimiter.

Furthermore, if you want to specify a quotation mark as part of the delimited string, you must use two quotation marks for each quotation mark that you want to appear in the string. For example, Suffix= "**""**" causes Oracle RMU to use a delimiter of **"**.

Include=Action=include−type

Specifies if deleted or modified records or transaction commit information is to be extracted from the after−image journal. The following keywords can be specified:

◊ Commit
NoCommit
If you specify Commit, a transaction commit record is written to each output stream as the final record for each transaction. The commit information record is written to output streams after all other records for the transaction have been written. The default is NoCommit.
Because output streams are created with a default file name of the table being extracted, it is important to specify a unique file name on each occurrence of the output stream. The definition of "unique" is such that when you write to a non−file−oriented output device (such as a pipe or mailbox), you must be certain to specify a specific file name on each output destination. This means that rather than specifying *Output=MBA1234:* for each output stream, you should use *Output=MBA1234:MBX*, or any file name that is the same on all occurrences of MBA1234:.
Failure to use a specific file name can result in additional, and unexpected, commit records being returned. However, this is generally a restriction only when using a stream−oriented output device (as opposed to a disk file).
The binary record format is based on the standard LogMiner output format. However, some fields are not used in the commit action record. The binary format and contents of this record are shown in Table 7−2. This record type is written for all output data formats.

Table 7−2 Commit Record Contents

Field	Length (in bytes)	Contents
ACTION	1	"C"
RELATION	31	Zero
RECORD_TYPE	4	Zero
DATA_LEN	2	Length of RM_TID_LEN, AERCP_LEN, RM_TID, AERCP
NBV_LEN	2	Zero
LDBK	8	Zero
START_TAD	8	Transaction Start Time/Date
COMMIT_TAD	8	Transaction Commit Time/Date
TSN	8	Transaction ID
RM_TID_LEN	4	Length of the Global TID
AERCP_LEN	4	Length of the AERCP information
RM_TID	RM_TID_LEN	Global TID
AERCP	AERCP_LEN	Restart Control Information

When the original transaction took part in a distributed, two−phase transaction, the RM_TID component is the Global transaction manager (XA or DDTM) unique transaction ID. Otherwise, this field contains binary zeroes.

The AIJ Extract Recovery Control Point (AERCP) information is used to uniquely identify this transaction within the scope of the database and after−image journal files. It contains the .aij sequence number, VBN and TSN of the last "Micro Quiet Point", and is used by the Continuous LogMiner process to restart a particular point in the journal sequence.

◊ Delete
NoDelete
If you specify Delete, pre−deletion record contents are extracted from the aij file. If you specify NoDelete, no pre−deletion record contents are extracted. The default is Delete.

◊ Modify
NoModify
If you specify Modify, modified or added record contents are extracted from the .aij file. If you specify NoModify, then no modified or added record contents are extracted. The default is Modify.

IO_Buffers=integer

Specifies the number of I/O buffers used for output data files. The default number of buffers is two, which is generally adequate. With sufficiently fast I/O subsystem hardware, additional buffers may improve performance. However, using a larger number of buffers will also consume additional virtual memory and process working set.

Log

Nolog

Specifies that the extraction of the .aij file is be reported to SYS$OUTPUT or the destination specified with the Output qualifier. When activity is logged, the output from the Log qualifier provides the number of transactions committed or rolled back. The default is the setting of the DCL VERIFY flag, which is controlled by the DCL SET VERIFY command.

Options=options−list

The following options can be specified:

◊ File=file−spec
An options file contains a list of tables and output destinations. The options file can be used instead of, or along with, the Table qualifier to specify the tables to be extracted. Each line of the options file must specify a table name prefixed with "Table=". After the table name, the output destination is specified as either "Output=", or "Callback_Module=" and "Callback_Routine=", for example:

```
TABLE=tblname,OUTPUT=outfile
TABLE=tblname,CALLBACK_MODULE=image,CALLBACK_ROUTINE=routine
```

You can use the Record_Definition=file−spec option from the Table qualifier to create a record definition file for the output data. The default file type is .rrd; the default file name is the name of the table.
You can use the Table_Definition=file−spec option from the Table qualifier to create a file that contains an SQL statement that creates a table to hold

transaction data. The default file type is .sql; the default file name is the name of the table.

Each option in the Options=File qualifier must be fully specified (no abbreviations are allowed) and followed with an equal sign (=) and a value string. The value string must be followed by a comma or the end of a line. Continuation lines can be specified by using a trailing dash. Comments are indicated by using the exclamation point (!) character.

You can use the asterisk (*) and the percent sign (%) wildcard characters in the table name specification to select all tables that satisfy the components you specify. The asterisk matches zero or more characters; the percent sign matches a single character.

For table name specifications that contain wild card characters, if the first character of the string is a pound sign (#), the wildcard specification is changed to a "not matching" comparison. This allows exclusion of tables based on a wildcard specification. The pound sign designation is only evaluated when the table name specification contains an asterisk or percent sign.

For example, a table name specification of "#FOO%" indicates that all table names that are four characters long and do *not* start with the string "FOO" are to be selected.

◊ Shared_Read

Specifies that the input after−image journal backup files are to be opened with an RMS shared locking specification.

◊ Dump

Specifies that the contents of an input metadata file are to be formatted and displayed. Typically, this information is used as a debugging tool.

Order_AIJ_Files

NoOrder_AIJ_Files

By default, after−image journal files are processed in the order that they are presented to the RMU Unload After_Journal command. The Order_AIJ_Files qualifier specifies that the input after−image journal files are to be processed in increasing order by sequence number. This can be of benefit when you use wildcard (* or %) processing of a number of input files. The .aij files are each opened, the first block is read (to determine the sequence number), and the files are closed prior to the sorting operation.

Output=file−spec

Redirects the log and trace output (selected with the Log and Trace qualifiers) to the named file. If this qualifier is not specified, the output generated by the Log and Trace qualifiers, which can be voluminous, is displayed to SYS$OUTPUT.

Parameter=character−strings

Specifies one or more character strings that are concatenated together and passed to the callback routine upon startup.

For each table that is associated with a user−supplied callback routine, the callback routine is called with two parameters: the length of the Parameter record and a pointer to the Parameter record. The binary format and contents of this record are shown in Table 7−3.

Table 7−3 Parameter Record Contents

Field	Length (in bytes)	Contents
ACTION	1	"P"
RELATION	31	Relation name
RECORD_TYPE	4	Zero
DATA_LEN	2	Length of parameter string
NBV_LEN	2	Zero
LDBK	8	Zero
START_TAD	8	Zero
COMMIT_TAD	8	Zero
TSN	8	Zero
DATA	?	Variable length parameter string content

Restart=restart−point

Specifies an AIJ Extract Restart Control Point (AERCP) that indicates the location to begin the extraction. The AERCP indicates the transaction sequence number (TSN) of the last extracted transaction along with a location in the .aij file where a known "Micro−quiet point" exists.

When the Restart qualifier is not specified and no input after−image journal files are specified on the command line, the Continuous LogMiner process starts extracting at the beginning of the earliest modified online after−image journal file.

When formatted for text display, the AERCP structure consists of the six fields (the MBZ field is excluded) displayed as unsigned integers separated by dashes; for example, "1−28−12−7−3202−3202".

Restore_Metadata=file−spec

Specifies that the RMU Unload After_Journal command is to read database metadata information from the specified file. The Database parameter is required but the database itself is not accessed when the Restore_Metadata qualifier is specified. The default file type is .metadata. The Continuous qualifier is not allowed when the Restore_Metadata qualifier is present.

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

When the Restore_Metadata qualifier is specified, additional checks are made to ensure that the after−image journal files were created using the same database that was used to create the metadata file. These checks provide additional security and help prevent accidental mismatching of files.

Save_Metadata=file−spec

Specifies that the RMU Unload After_Journal command is to write metadata information to the named file. The Continuous, Restore_Metadata, Table, and

Options=file qualifiers and the aij−file−name parameter are not allowed when the Save_Metadata qualifier is present. The default file type is .metadata.

Select=selection−type

Specifies if the date and time of the Before and Since qualifiers refer to transaction start time or transaction commit time.

The following options can be specified as the selection−type of the Select qualifier:

- ◊ Commit_Transaction
  Specifies that the Before and Since qualifiers select transactions based on the time of the transaction commit.
- ◊ Start_Transaction
  Specifies that the Before and Since qualifiers select transactions based on the time of the transaction start.

The default for date selection is Commit_Transaction.

Since=date−time

Specifies the starting time for transactions to be extracted. Depending on the value specified in the Select qualifier, transactions that committed or started on or after the specified Since date are selected. Information from transactions that committed or started prior to the specified Since date is not included in the output.

Sort_Workfiles=integer

Specifies the number of sort work files. The default number of sort work files is two. When large transactions are being extracted, using additional sort work files may improve performance by distributing I/O loads over multiple disk devices. Use the SORTWORKn (where n is a number from 0 to 9) logical names to specify the location of the sort work files.

Statistics_Interval=integer

Specifies that statistics are to be displayed at regular intervals so that you can evaluate the progress of the unload operation.

The displayed statistics include:

- ◊ Elapsed time
- ◊ CPU time
- ◊ Buffered I/O
- ◊ Direct I/O
- ◊ Page faults
- ◊ Number of records unloaded for a table
- ◊ Total number of records extracted for all tables

If the Statistics_Interval qualifier is specified, the default interval is 60 seconds. The minimum value is one second. If the unload operation completes successfully before the first time interval has passed, you will receive an informational message on the number of files unloaded. If the unload operation is unsuccessful before the first time interval has passed, you will receive error messages and statistics on the number of records unloaded.

At any time during the unload operation, you can press Ctrl/T to display the current statistics.

Table=(Name=table−name, table−options)

Specifies the name of a table to be unloaded and an output destination. The table−name must be a table within the database. Views, indexes, and system relations may not be unloaded from the after−image journal file.

The asterisk (*) and the percent sign (%) wildcard characters can be used in the table name specification to select all tables that satisfy the components you specify. The asterisk matches zero or more characters and the percent sign matches a single character.

For table name specifications that contain wild card characters, if the first character of the string is a pound sign (#), the wildcard specification is changed to a "not matching" comparison. This allows exclusion of tables based on a wildcard specification. The pound sign designation is only evaluated when the table name specification contains an asterisk or percent sign.

For exmple, a table name specification of "#FOO%" indicates that all table names that are four characters long and do *not* start with the string "FOO" are to be selected.

The following table−options can be specified with the Table qualifier:

- ◊ Output=file−spec
  If an Output file specification is present, unloaded records are written to the specified location.
- ◊ Callback_Module=image−name, Callback_Routine=routine−name
  The LogMiner process uses the OpenVMS library routine LIB$FIND_IMAGE_SYMBOL to activate the specified shareable image and locate the specified entry point routine name. This routine is called with each extracted record. A final call is made with the Action field set to "E" to indicate the end of the output stream. These options must be specified together.
- ◊ Record_Definition=file−spec
  The Record_Definition=file−spec option can be used to create a record definition file for the output data. The default file type is .rrd; the default file name is the name of the table.
- ◊ Table_Definition=file−spec
  You can use the Table_Definition=file−spec option to create a file that contains an SQL statement that creates a table to hold transaction data. The default file type is .sql; the default file name is the name of the table.

Unlike other qualifiers where only the final occurrence of the qualifier is used by an application, the Table qualifier can be specified multiple times for the RMU Unload After_Journal command. Each occurrence of the Table qualifier must specify a different table.

Trace

NoTrace

Specifies that the unloading of the .aij file be traced. The default is Notrace. When the

unload operation is traced, the output from the Trace qualifier identifies transactions in the .aij file by TSNs and describes what Oracle RMU did with each transaction during the unload process. You can specify the Log qualifier with the Trace qualifier.

USAGE NOTES

◊ To use the RMU Unload After_Journal command for a database, you must have the RMU$DUMP privilege in the root file access control list (ACL) for the database or the OpenVMS SYSPRV or BYPASS privilege.

◊ When the Continuous qualifier is not specified, you can only extract changed records from a backup copy of the after−image journal files. You create this file using the RMU Backup After_Journal command.
You cannot extract from an .aij file that has been optimized with the RMU Optimize After_Journal command.

◊ As part of the extraction process, Oracle RMU sorts extracted journal records to remove duplicate record updates. Because .aij file extraction uses the OpenVMS Sort/Merge Utility (SORT/MERGE) to sort journal records for large transactions, you can improve the efficiency of the sort operation by changing the number and location of the work files used by SORT/MERGE. The number of work files is controlled by the Sort_Workfiles qualifier of the RMU Unload After_Journal command. The allowed values are 1 through 10 inclusive, with a default value of 2. The location of these work files can be specified with device specifications, using the SORTWORKn logical name (where n is a number from 0 to 9). See the OpenVMS documentation set for more information on using SORT/MERGE. See the *Oracle Rdb7 Guide to Database Performance and Tuning* for more information on using these Oracle Rdb logical names.

◊ When extracting large transactions, the RMU Unload After_Journal command may create temporary work files. You can redirect the .aij rollforward temporary work files to a different disk and directory location than the current default directory by assigning a different directory to the RDM$BIND_AIJ_WORK_FILE logical name in the LNM$FILE_DEV name table. This can help to alleviate I/O bottlenecks that might occur on the default disk.

◊ The RMU Unload After_Journal command can read either a backed up .aij file on disk or a backed up .aij file on tape that is in the Old_File format.

◊ You can select one or more tables to be extracted from an after−image journal file. All tables specified by the Table qualifier and all those specified in the Options file are combined to produce a single list of output streams. A particular table can be specified only once. Multiple tables can be written to the same output destination by specifying the exact same output stream specification (that is, by using an identical file specification).

◊ At the completion of the unload operation, RMU creates a number of DCL symbols that contain information about the extraction statistics. For each table extracted, RMU creates the following symbols:

- RMU$UNLOAD_DELETE_COUNT_tablename
- RMU$UNLOAD_MODIFY_COUNT_tablename
- RMU$UNLOAD_OUTPUT_tablename

The tablename component of the symbol is the name of the table. When multiple tables are extracted in one operation, multiple sets of symbols are created. The value for the symbols RMU$UNLOAD_MODIFY_COUNT_tablename and

RMU$UNLOAD_DELETE_COUNT_tablename is a character string containing the number of records returned for modified and deleted rows. The RMU$UNLOAD_OUTPUT_tablename symbol is a character string indicating the full file specification for the output destination, or the shareable image name and routine name when the output destination is an application callback routine.

◊ When you use the Callback_Module and Callback_Routine option, you must supply a shareable image with a universal symbol or entry point for the LogMiner process to be able to call your routine. See the OpenVMS documentation discussing the Linker utility for more information about creating shareable images.

◊ Your Callback_Routine is called once for each output record. The Callback_Routine is passed two parameters:

  · The length of the output record, by longword value
  · A pointer to the record buffer

The record buffer is a data structure of the same fields and lengths written to an output destination.

◊ Because the Oracle RMU image is installed as a known image, your shareable image must also be a known image. Use the OpenVMS Install Utility to make your shareable image known. You may wish to establish an exit handler to perform any required cleanup processing at the end of the extraction.

◊ Segmented string data (BLOB) cannot be extracted using the LogMiner process. Because the segmented string data is related to the base table row by means of a database key, there is no convenient way to determine what data to extract. Additionally, the data type of an extracted column is changed from LIST OF BYTE VARYING to BIGINT. This column contains the DBKEY of the original BLOB data. Therefore, the contents of this column should be considered unreliable. However, the field definition itself is extracted as a quadword integer representing the database key of the original segmented string data. In generated table definition or record definition files, a comment is added indicating that the segmented string data type is not supported by the LogMiner for Rdb feature.

◊ Records removed from tables using the SQL TRUNCATE TABLE statement are not extracted. The SQL TRUNCATE TABLE statement does not journal each individual data record being removed from the database.

◊ Records removed from tables using the SQL ALTER DATABASE command with the DROP STORAGE AREA clause and CASCADE keyword are not extracted. Any data deleted by this process is not journalled.

◊ Records removed by dropping tables using the SQL DROP TABLE statement are not extracted. The SQL DROP TABLE statement does not journal each individual data record being removed from the database.

◊ When the RDMS$CREATE_LAREA_NOLOGGING logical is defined, DML operations are not available for extraction between the time the table is created and when the transaction is committed.

◊ Tables that use the vertical record partitioning (VRP) feature cannot be extracted using the LogMiner feature. LogMiner software currently does not detect these tables. A future release of Oracle Rdb will detect and reject access to vertically partitioned tables.

◊ In binary format output, VARCHAR fields are not padded with spaces in the output file. The VARCHAR data type is extracted as a 2−byte count field and a fixed−length data field. The 2−byte count field indicates the number of valid characters in the fixed−length data field. Any additional contents in the

data field are unpredictable.

◊ You cannot extract changes to a table when the table definition is changed within an after−image journal file. Data definition language (DDL) changes to a table are not allowed within an .aij file being extracted. All records in an .aij file must be the current record version. If you are going to perform DDL operations on tables that you wish to extract using the LogMiner for Rdb, you should:

1. Back up your after−image journal files.
2. Extract the .aij files using the RMU Unload After_Journal command.
3. Make the DDL changes.

◊ Do not use the OpenVMS Alpha High Performance Sort/Merge utility (selected by defining the logical name SORTSHR to SYS$SHARE:HYPERSORT) when using the LogMiner feature. HYPERSORT supports only a subset of the library sort routines that LogMiner requires. Make sure that the SORTSHR logical name is not defined to HYPERSORT.

◊ You can use an OpenVMS pipe to pass data from the RMU Unload After_Journal command to another application (for example, RMU Load). Do not use any options (such as the Log or Verify qualifiers) that could cause the LogMiner process to send extra output to the SYS$OUTPUT device, as that information would be part of the input data source stream to the next pipeline segment.

You may find that the OpenVMS default size of the pipe is too small if the records being extracted (including LogMiner fields) are larger than 256 bytes. If the pipe is too small, increase the SYSGEN parameters MAXBUF and DEFMBXMXMSG, and reboot the system.

The following example uses LogMiner for Rdb to direct output to an OpenVMS pipe device. It uses the RMU Load command to read the pipe device as the input data record stream. Using the pipeline allows parallel processing and also avoids the need for an intermediate disk file. You must create the record definition (.rrd) file prior to executing the command.

```
$ PIPE (RMU /UNLOAD /AFTER_JOURNAL OLTP.RDB AIJ1.AIJ −
    /TABLE = (NAME = MYTBL, OUTPUT = SYS$OUTPUT:)) −
     |  (RMU /LOAD REPORTS.RDB MYLOGTBL SYS$PIPE: −
        /RECORD_DEFINITION = FILE = MYLOGTBL.RRD)
```

USAGE NOTES FOR THE CONTINUOUS LOGMINER FEATURE

♦ You can specify input backup after−image journal files along with the Continuous qualifier from the command line. The specified after−image journal backup files are processed in an offline mode. Once they have been processed, the RMU Unload After_Journal command switches to "online" mode and the active online journals are processed.

♦ When no input after−image journal files are specified on the command line, the Continuous LogMiner starts extracting at the beginning of the earliest modified online after−image journal file. The Restart= qualifier can be used to control the first transaction to be extracted.

♦ The Continuous LogMiner requires fixed−size circular after−image journals.

♦ An after−image journal file cannot be backed up if there are any Continuous LogMiner checkpoints in the aij file. The Continuous LogMiner moves its checkpoint to the physical end−of−file for the online .aij file that it is extracting.

♦ In order to ensure that all records have been written by all database users, Continuous LogMiner processes do not switch to the next live journal file until it has been written

to by another process. Live journals should not be backed up while the Continuous LogMiner process is processing a list of .aij backup files.

- If backed up after−image journal files are specified on the command line and the Continuous qualifier is specified, the journal sequence numbers must ascend directly from the backed up journal files to the online journal files.
  In order to preserve the after−image journal file sequencing as processed by the RMU Unload After_Journal /Continuous command, it is important that no after−image journal backup operations are attempted between the start of the command and when the Continuous LogMiner process reaches the live online after−image journals.
- You can run multiple Continuous LogMiner processes at one time on a database. Each Continuous LogMiner process acts independently.
- The Continuous LogMiner reads the live after−image journal file just behind writers to the journal. This will likely increase the I/O load on the disk devices where the journals are located. The Continuous LogMiner attempts to minimize unneeded journal I/O by checking a "High Water Mark" lock to determine if the journal has been written to and where the highest written block location is located.
- Vertically partitioned tables cannot be extracted.

USING LOGMINER TO MINIMIZE APPLICATION DOWNTIME FOR MAINTENANCE

Lengthy offline application or database maintenance operations can pose a significant problem in high−availability production environments. The LogMiner for Rdb feature can help reduce the length of downtime to a matter of minutes.

If you use a backup of the database for maintenance operations, you can continue to modify the application during lengthy maintenance operations. Once the maintenance is complete, you can shut down the application, back up the production system .aij files, and use the LogMiner feature to extract changes made to production tables since the database was backed up. You can then use an application program or the trigger technique to apply the changes to the new database. Once the new database has been updated, you can restart the application using the new database.

The sequence of events required is similar to the following:

1. Perform a full online, quiet−point database backup of the production database.
2. Restore the backup to create a new database that will eventually become the production database.
3. Perform maintenance operations on the new database. The product system continues to run.
4. Perform an online, quiet−point after−image journal backup of the production database.
5. Use the RMU Unload After_Journal command to unload all database tables into individual output files from the .aij backup file.
6. Using either the trigger technique or an application program, update the tables in the new database with the changed data.
7. Shut down the production application and close the database.
8. Perform an offline, quiet−point after−image journal backup of the production database.
9. Use the RMU Unload After_Journal command to unload all database tables into individual output files from the .aij backup file.
10. Using either the trigger technique or an application program, update the tables in the new database with the changed data.

11. Start an online, quiet−point backup of the new database.
12. Change logical names or the environment to specify the new database root file as the production database.
13. Restart the application on the new database.

Depending on the amount of application database activity, steps 4, 5, and 6 can be repeated to limit the amount of data that needs to be applied and the amount of downtime required during the final after−image journal backup and apply stage in steps 8, 9, and 10.

EXAMPLES

Example 1

The following example unloads the EMPLOYEES table from the .aij backup file MFP.AIJBCK.

```
RMU /UNLOAD /AFTER_JOURNAL MFP.RDB MFP.AIJBCK −
    /TABLE = (NAME = EMPLOYEES, OUTPUT = EMPLOYEES.DAT)
```

Example 2

The following example simultaneously unloads the SALES, STOCK, SHIPPING, and ORDERS tables from the .aij backup files MFS.AIJBCK_1−JUL−1999 through MFS.AIJBCK_3−JUL−1999. Note that the input .aij backup files are processed sequentially in the order specified.

```
$ RMU /UNLOAD /AFTER_JOURNAL MFS.RDB −
   MFS.AIJBCK_1−JUL−1999, −
   MFS.AIJBCK_2−JUL−1999, −
   MFS.AIJBCK_3−JUL−1999 −
   /TABLE = (NAME = SALES, OUTPUT = SALES.DAT) −
   /TABLE = (NAME = STOCK, OUTPUT = STOCK.DAT) −
   /TABLE = (NAME = SHIPPING, OUTPUT = SHIPPING.DAT) −
   /TABLE = (NAME = ORDER, OUTPUT = ORDER.DAT)
```

Example 3

Use the Before and Since qualifiers to unload data based on a time range. The following example extracts changes made to the PLANETS table by transactions that committed between 1−SEP−1999 at 14:30 and 3−SEP−1999 at 16:00.

```
$ RMU /UNLOAD /AFTER_JOURNAL MFS.RDB MFS.AIJBCK −
   /TABLE = (NAME = PLANETS, OUTPUT = PLANETS.DAT) −
   /BEFORE = "3−SEP−1999 16:00:00.00" −
   /SINCE = "1−SEP−1999 14:30:00.00"
```

Example 4

The following example simultaneously unloads the SALES and STOCK tables from all .aij backup files that match the wildcard specification MFS.AIJBCK_1999−07−*. The input .aij backup files are processed sequentially in the order returned from the file system.

```
$ RMU /UNLOAD /AFTER_JOURNAL MFS.RDB −
   MFS.AIJBCK_1999−07−* −
   /TABLE = (NAME = SALES, OUTPUT = SALES.DAT) −
   /TABLE = (NAME = STOCK, OUTPUT = STOCK.DAT)
```

Example 5

The following example unloads the TICKER table from the .aij backup files listed in the file called AIJ_BACKUP_FILES.DAT (note the double quotation marks surrounding the at (@) character and the file specification). The input .aij backup files are processed sequentially. The output records are written to the mailbox device called MBA127:. A separate program is already running on the system, and it reads and processes the data written to the mailbox.

```
$ RMU /UNLOAD /AFTER_JOURNAL MFS.RDB -
   "@AIJ_BACKUP_FILES.DAT" -
   /TABLE = (NAME = TICKER, OUTPUT = MBA127:)
```

Example 6

You can use the RMU Unload After_Journal command followed by RMU Load commands to move transaction data from one database into a change table in another database. You must create a record definition (.rrd) file for each table being loaded into the target database. The record definition files can be created by specifying the Record_Definition option on the Table qualifier.

```
$ RMU /UNLOAD /AFTER_JOURNAL OLTP.RDB MYAIJ.AIJBCK -
  /TABLE = ( NAME = MYTBL, -
             OUTPUT = MYTBL.DAT, -
             RECORD_DEFINITION=MYLOGTBL) -
  /TABLE = ( NAME = SALE, -
             OUTPUT=SALE.DAT, -
             RECORD_DEFINITION=SALELOGTBL)

$ RMU /LOAD WAREHOUSE.RDB MYLOGTBL MYTBL.DAT -
   /RECORD_DEFINITION = FILE = MYLOGTBL.RRD

$ RMU /LOAD WAREHOUSE.RDB SALELOGTBL SALE.DAT -
   /RECORD_DEFINITION = FILE = SALELOGTBL.RRD
```

Example 7

You can use an RMS file containing the record structure definition for the output file as an input file to the RMU Load command. The record description uses the CDO record and field definition format. This is the same format used by the RMU Load and RMU Unload commands when the Record_Definition qualifier is used. The default file extension is .rrd.

The record definitions for the fields that the LogMiner processs writes to the output .rrd file are shown in the following table. These fields can be manually appended to a record definition file for the actual user data fields being unloaded. The file can be used to load a *transaction table* within a database. A transaction table is the output that the LogMiner process writes to a table consisting of sequential transactions performed in a database.

```
DEFINE FIELD RDB$LM_ACTION          DATATYPE IS TEXT SIZE IS 1.
DEFINE FIELD RDB$LM_RELATION_NAME   DATATYPE IS TEXT SIZE IS 31.
DEFINE FIELD RDB$LM_RECORD_TYPE     DATATYPE IS SIGNED LONGWORD.
DEFINE FIELD RDB$LM_DATA_LEN        DATATYPE IS SIGNED WORD.
DEFINE FIELD RDB$LM_NBV_LEN         DATATYPE IS SIGNED WORD.
DEFINE FIELD RDB$LM_DBK             DATATYPE IS SIGNED QUADWORD.
DEFINE FIELD RDB$LM_START_TAD       DATETYPE IS DATE
DEFINE FIELD RDB$LM_COMMIT_TAD      DATATYPE IS DATE
DEFINE FIELD RDB$LM_TSN             DATATYPE IS SIGNED QUADWORD.
DEFINE FIELD RDB$LM_RECORD_VERSION  DATATYPE IS SIGNED WORD.
```

Example 8

Instead of using the Table qualifier, you can use an Options file to specify the table or tables to be extracted, as shown in the following example.

```
$ TYPE TABLES.OPTIONS
TABLE=MYTBL, OUTPUT=MYTBL.DAT
TABLE=SALES, OUTPUT=SALES.DAT
$ RMU /UNLOAD /AFTER_JOURNAL OLTP.RDB MYAIJ.AIJBCK -
   /OPTIONS = FILE = TABLES.OPTIONS
```

Example 9

The following example unloads the EMPLOYEES table from the live database and writes all change records to the MBA145 device. A separate program is presumed to be reading the mailbox at all times and processing the records.

```
$ RMU /UNLOAD /AFTER_JOURNAL /CONTINUOUS MFP.RDB -
 /TABLE = (NAME = EMPLOYEES, OUTPUT = MBA145:)
```

Example 10

This example demonstrates unloading three tables (EMPLOYEES, SALES, and CUSTOMERS) to a single mailbox. Even though the mailbox is not a file−oriented device, the same file name is specified for each. This is required because the LogMiner process defaults the file name to the table name. If the same file name is not explicitly specified for each output stream destination, the LogMiner process assigns one mailbox channel for each table. When the file name is the same for all tables, the LogMiner process detects this and assigns only a single channel for all input tables.

```
$ DEFINE MBX$ LOADER_MBX:X
$ RMU /UNLOAD /AFTER_JOURNAL /CONTINUOUS MFP.RDB -
 /TABLE = (NAME = EMPLOYEES, OUTPUT = MBX$:) -
 /TABLE = (NAME = SALES, OUTPUT = MBX$:) -
 /TABLE = (NAME = CUSTOMERS, OUTPUT = MBX$:)
```

Example 11

In order to include transaction commit information, the /Include =Action =Commit qualifier is specified in this example. Additionally, the EMPLOYEES and SALES tables are extracted to two different mailbox devices (ready by separate processes). A commit record is written to each mailbox after all changed records for each transaction have been extracted.

```
$ RMU /UNLOAD /AFTER_JOURNAL /CONTINUOUS MFP.RDB -
 /INCLUDE = ACTION = COMMIT -
 /TABLE = (NAME = EMPLOYEES, OUTPUT = LOADER_EMP_MBX:X) -
 /TABLE = (NAME = SALES, OUTPUT = LOADER_SAL_MBX:X)
```

Example 12

In this example, multiple input backup after−image journal files are supplied. The Order_AIJ_Files qualifier specifies that the .aij files are to be processed in ascending order of .aij sequence number (regardless of file name). Prior to the extraction operation, each input file is opened and the .aij Open record is read. The .aij files are then opened and extracted, one at a time, by ascending .aij sequence number.

```
$ RMU /UNLOAD /AFTER_JOURNAL /LOG /ORDER_AIJ_FILES -
 MFP.RDB *.AIJBCK -
 /TABLE = (NAME = C1, OUTPUT=C1.DAT)
%RMU-I-UNLAIJFL, Unloading table C1 to DGA0:[DB]C1.DAT;1
%RMU-I-LOGOPNAIJ, opened journal file DGA0:[DB]ABLE.AIJBCK;1
%RMU-I-AIJRSTSEQ, journal sequence number is "5"
%RMU-I-LOGOPNAIJ, opened journal file DGA0:[DB]BAKER.AIJBCK;1
%RMU-I-AIJRSTSEQ, journal sequence number is "4"
%RMU-I-LOGOPNAIJ, opened journal file DGA0:[DB]CHARLIE.AIJBCK;1
%RMU-I-AIJRSTSEQ, journal sequence number is "6"
%RMU-I-LOGOPNAIJ, opened journal file DGA0:[DB]BAKER.AIJBCK;1
%RMU-I-AIJRSTSEQ, journal sequence number is "4"
%RMU-I-AIJMODSEQ, next AIJ file sequence number will be 5
%RMU-I-LOGOPNAIJ, opened journal file DGA0:[DB]ABLE.AIJBCK;1
%RMU-I-AIJRSTSEQ, journal sequence number is "5"
%RMU-I-AIJMODSEQ, next AIJ file sequence number will be 6
%RMU-I-LOGOPNAIJ, opened journal file DGA0:[DB]CHARLIE.AIJBCK;1
%RMU-I-AIJRSTSEQ, journal sequence number is "6"
%RMU-I-AIJMODSEQ, next AIJ file sequence number will be 7
%RMU-I-LOGSUMMARY, total 7 transactions committed
%RMU-I-LOGSUMMARY, total 0 transactions rolled back
------------------------------------------------------------------------------
 ELAPSED: 0 00:00:00.15 CPU: 0:00:00.08 BUFIO: 62 DIRIO: 19 FAULTS: 73
Table "C1" : 3 records written (3 modify, 0 delete)
Total : 3 records written (3 modify, 0 delete)
```

Example 13

The SQL record definitions for the fields that the LogMiner process writes to the output are shown in the following example. These fields can be manually appended to the table creation command for the actual user data fields being unloaded. Alternately, the Table_Definition qualifier can be used with the Table qualifier or within an Options file to automatically create the SQL definition file. This can be used to create a transaction table of changed data.

```
SQL> create table MYLOGTABLE (
cont> RDB$LM_ACTION          CHAR,
cont> RDB$LM_RELATION_NAME   CHAR (31),
cont> RDB$LM_RECORD_TYPE     INTEGER,
cont> RDB$LM_DATA_LEN        SMALLINT,
cont> RDB$LM_NBV_LEN         SMALLINT,
cont> RDB$LM_DBK             BIGINT,
cont> RDB$LM_START_TAD       DATE VMS,
cont> RDB$LM_COMMIT_TAD      DATE VMS,
cont> RDB$LM_TSN             BIGINT,
cont> RDB$LM_RECORD_VERSION  SMALLINT ...);
```

Example 14

The following example is the transaction table record definition (.rrd) file for the EMPLOYEES table from the PERSONNEL database:

```
DEFINE FIELD RDB$LM_ACTION          DATATYPE IS TEXT SIZE IS 1.
DEFINE FIELD RDB$LM_RELATION_NAME   DATATYPE IS TEXT SIZE IS 31.
DEFINE FIELD RDB$LM_RECORD_TYPE     DATATYPE IS SIGNED LONGWORD.
DEFINE FIELD RDB$LM_DATA_LEN        DATATYPE IS SIGNED WORD.
DEFINE FIELD RDB$LM_NBV_LEN         DATATYPE IS SIGNED WORD.
DEFINE FIELD RDB$LM_DBK             DATATYPE IS SIGNED QUADWORD.
DEFINE FIELD RDB$LM_START_TAD       DATATYPE IS DATE.
DEFINE FIELD RDB$LM_COMMIT_TAD      DATATYPE IS DATE.
DEFINE FIELD RDB$LM_TSN             DATATYPE IS SIGNED QUADWORD.
DEFINE FIELD RDB$LM_RECORD_VERSION  DATATYPE IS SIGNED WORD.
```

```
DEFINE FIELD EMPLOYEE_ID            DATATYPE IS TEXT SIZE IS 5.
DEFINE FIELD LAST_NAME              DATATYPE IS TEXT SIZE IS 14.
DEFINE FIELD FIRST_NAME             DATATYPE IS TEXT SIZE IS 10.
DEFINE FIELD MIDDLE_INITIAL         DATATYPE IS TEXT SIZE IS 1.
DEFINE FIELD ADDRESS_DATA_1         DATATYPE IS TEXT SIZE IS 25.
DEFINE FIELD ADDRESS_DATA_2         DATATYPE IS TEXT SIZE IS 20.
DEFINE FIELD CITY                   DATATYPE IS TEXT SIZE IS 20.
DEFINE FIELD STATE                  DATATYPE IS TEXT SIZE IS 2.
DEFINE FIELD POSTAL_CODE            DATATYPE IS TEXT SIZE IS 5.
DEFINE FIELD SEX                    DATATYPE IS TEXT SIZE IS 1.
DEFINE FIELD BIRTHDAY               DATATYPE IS DATE.
DEFINE FIELD STATUS_CODE            DATATYPE IS TEXT SIZE IS 1.

DEFINE RECORD EMPLOYEES.
   RDB$LM_ACTION .
   RDB$LM_RELATION_NAME .
   RDB$LM_RECORD_TYPE .
   RDB$LM_DATA_LEN .
   RDB$LM_NBV_LEN .
   RDB$LM_DBK .
   RDB$LM_START_TAD .
   RDB$LM_COMMIT_TAD .
   RDB$LM_TSN .
   RDB$LM_RECORD_VERSION .
   EMPLOYEE_ID .
   LAST_NAME .
   FIRST_NAME .
   MIDDLE_INITIAL .
   ADDRESS_DATA_1 .
   ADDRESS_DATA_2 .
   CITY .
   STATE .
   POSTAL_CODE .
   SEX .
   BIRTHDAY .
   STATUS_CODE .
END EMPLOYEES RECORD.
```

Example 15

The following C source code segment demonstrates the structure of a module that can be used as a callback module and routine to process employee transaction information from the LogMiner process. The routine, Employees_Callback, would be called by the LogMiner process for each extracted record. The final time the callback routine is called, the RDB$LM_ACTION field will be set to "E" to indicate the end of the output stream.

```
#include <stdio>
typedef unsigned char date_type[8];
typedef unsigned char dbkey_type[8];
typedef unsigned char tsn_type[8];

typedef struct {
    unsigned char       rdb$lm_action;
    char                rdb$lm_relation_name[31];
    unsigned int        rdb$lm_record_type;
    unsigned short int  rdb$lm_data_len;
    unsigned short int  rdb$lm_nbv_len;
    dbkey_type          rdb$lm_dbk;
    date_type           rdb$lm_start_tad;
    date_type           rdb$lm_commit_tad;
    tsn_type            rdb$lm_tsn;
    unsigned short int  rdb$lm_record_version;
    char                employee_id[5];
```

```
    char                last_name[14];
    char                first_name[10];
    char                middle_initial[1];
    char                address_data_1[25];
    char                address_data_2[20];
    char                city[20];
    char                state[2];
    char                postal_code[5];
    char                sex[1];
    date_type           birthday;
    char                status_code[1];
} transaction_data;

void employees_callback (unsigned int data_len, transaction_data data_buf)
{    .
     .
     .
 return;}
```

Use the C compiler (either :VAX C or DEC C) to compile this module. When linking this module, the symbol EMPLOYEES_CALLBACK needs to be externalized in the shareable image. Refer to the OpenVMS manual discussing the Linker utility for more information about creating shareable images.

On OpenVMS Alpha systems, you can use a LINK command similar to the following:

```
$ LINK /SHAREABLE = EXAMPLE.EXE EXAMPLE.OBJ + SYS$INPUT: /OPTIONS
SYMBOL_VECTOR = (EMPLOYEES_CALLBACK = PROCEDURE)
<Ctrl/Z>
```

On OpenVMS VAX systems, you can use a LINK command similar to the following:

```
$ LINK /SHAREABLE = EXAMPLE.EXE EXAMPLE.OBJ + SYS$INPUT: /OPTIONS
UNIVERSAL = EMPLOYEES_CALLBACK
<Ctrl/Z>
```

Example 16

You can use triggers and a transaction table to construct a method to replicate table data from one database to another using RMU Unload After_Journal and RMU Load commands. This data replication method is based on transactional changes to the source table and requires no programming. Instead, existing features of Oracle Rdb can be combined to provide this functionality.

For this example, consider a simple customer information table called CUST with a unique customer ID value, customer name, address, and postal code. Changes to this table are to be moved from an OLTP database to a reporting database system on a periodic (perhaps nightly) basis.

First, in the reporting database, a customer table of the same structure as the OLTP customer table is created. In this example, this table is called RPT_CUST. It contains the same fields as the OLTP customer table called CUST.

```
SQL> CREATE TABLE RPT_CUST
cont> CUST_ID               INTEGER,
cont> CUST_NAME             CHAR (50),
cont> CUST_ADDRESS          CHAR (50),
cont> CUST_POSTAL_CODE      INTEGER);
```

Next, a temporary table is created in the reporting database for the LogMiner−extracted transaction data from the CUST table. This temporary table definition specifies ON COMMIT DELETE ROWS so that data in the temporary table is deleted from memory at each transaction commit. A temporary table is used because there is no need to journal changes to the table.

```
SQL> CREATE GLOBAL TEMPORARY TABLE RDB_LM_RPT_CUST (
cont> RDB$LM_RECORD_TYPE    INTEGER,
cont> RDB$LM_DATA_LEN       SMALLINT,
cont> RDB$LM_NBV_LEN        SMALLINT,
cont> RDB$LM_DBK            BIGINT,
cont> RDB$LM_START_TAD      DATE VMS,
cont> RDB$LM_COMMIT_TAD     DATE VMS,
cont> RDB$LM_TSN            BIGINT,
cont> RDB$LM_RECORD_VERSION SMALLINT,
cont> CUST_ID               INTEGER,
cont> CUST_NAME             CHAR (50),
cont> CUST_ADDRESS          CHAR (50),
cont> CUST_POSTAL_CODE      INTEGER) ON COMMIT DELETE ROWS;
```

For data to be populated in the RPT_CUST table in the reporting database, a trigger is created for the RDB_LM_RPT_CUST transaction table. This trigger is used to insert, update, or delete rows in the RPT_CUST table based on the transaction information from the OLTP database for the CUST table. The unique CUST_ID field is used to determine if customer records are to be modified or added.

```
SQL> CREATE TRIGGER RDB_LM_RPT_CUST_TRIG
cont>  AFTER INSERT ON RDB_LM_RPT_CUST
cont>
cont> -- Modify an existing customer record
cont>
cont>  WHEN (RDB$LM_ACTION = 'M' AND
cont>        EXISTS (SELECT RPT_CUST.CUST_ID FROM RPT_CUST
cont>                WHERE RPT_CUST.CUST_ID = RDB_LM_RPT_CUST.CUST_ID))
cont>      (UPDATE RPT_CUST SET
cont>              RPT_CUST.CUST_NAME = RDB_LM_RPT_CUST.CUST_NAME,
cont>              RPT_CUST.CUST_ADDRESS = RDB_LM_RPT_CUST.CUST_ADDRESS,
cont>              RPT_CUST.CUST_POSTAL_CODE = RDB_LM_RPT_CUST.CUST_POSTAL_CODE
cont>       WHERE RPT_CUST.CUST_ID = RDB_LM_RPT_CUST.CUST_ID)
cont>  FOR EACH ROW
cont>
cont> -- Add a new customer record
cont>
cont>  WHEN (RDB$LM_ACTION = 'M' AND NOT
cont>        EXISTS (SELECT RPT_CUST.CUST_ID FROM RPT_CUST
cont>                WHERE RPT_CUST.CUST_ID = RDB_LM_RPT_CUST.CUST_ID))
cont>      (INSERT INTO RPT_CUST VALUES
cont>              (RDB_LM_RPT_CUST.CUST_ID,
cont>               RDB_LM_RPT_CUST.CUST_NAME,
cont>               RDB_LM_RPT_CUST.CUST_ADDRESS,
cont>               RDB_LM_RPT_CUST.CUST_POSTAL_CODE))
cont>  FOR EACH ROW
cont>
cont> -- Delete an existing customer record
cont>
cont>  WHEN (RDB$LM_ACTION = 'D')
cont>      (DELETE FROM RPT_CUST
cont>       WHERE RPT_CUST.CUST_ID = RDB_LM_RPT_CUST.CUST_ID)
cont>  FOR EACH ROW;
```

Within the trigger, the action to take (for example, to add, update, or delete a customer record) is based on the RDB$LM_ACTION field (defined as D or M) and the existence of the

customer record in the reporting database. For modifications, if the customer record does not exist, it is added; if it does exist, it is updated. For a deletion on the OLTP database, the customer record is deleted from the reporting database.

The RMU Load command is used to read the output from the LogMiner process and load the data into the temporary table where each insert causes the trigger to execute. The Commit_Every qualifier is used to avoid filling memory with the customer records in the temporary table because as soon as the trigger executes, the record in the temporary table is no longer needed.

```
$ RMU /UNLOAD /AFTER_JOURNAL OLTP.RDB OLTP.AIJBCK −
 /TABLE = (NAME = CUST, −
           OUTPUT = CUST.DAT, −
           RECORD_DEFINITION = RDB_LM_RPT_CUST.RRD)

$ RMU /LOAD REPORT_DATABASE.RDB RDB_LM_RPT_CUST CUST.DAT −
      /RECORD_DEFINITION = FILE = RDB_LM_RPT_CUST.RRD −
      /COMMIT_EVERY = 1000
```

7.2 RMU Set Logminer Command

Format

RMU/Set Logminer root–file–spec

Command Qualifiers	Defaults
/Continuous	/NoContinuous
/Disable	See description
/Enable	See description
/[No]Log	Current DCL verify value

DESCRIPTION

Use this command to enable or disable LogMiner operations on an Oracle Rdb database. When LogMiner is enabled, the Oracle Rdb database software writes additional information to the after−image journal file when records are added, modified, and deleted from the database. This information is used during the unload operation.

COMMAND PARAMETERS

root–file–spec

The root file specification of the database. The default file extension is .rdb.

COMMAND QUALIFIERS

Continuous

NoContinuous

Enables the database for the Continuous LogMiner feature when used in conjunction with the Enable qualifier. Use the NoContinuous qualifier with the Enable qualifier to disable use of the Continuous LogMiner feature.

The RMU Set Logminer /Disable command explicitly disables the Continuous LogMiner feature as well as the base LogMiner functionality. To enable the Continuous LogMiner feature again, the entire RMU Set Logminer /Enable /Continuous command must be used.

Disable

Specifies that LogMiner operations are to be disabled for the database. When LogMiner is disabled, the Oracle Rdb software does not journal information required for LogMiner operations. When LogMiner is disabled for a database, the RMU Unload After_Journal command is not functional on that database.

Enable

Specifies that LogMiner operations are to be enabled for the database. When LogMiner is enabled, the Oracle Rdb database software logs additional information to the after−image journal file. This information allows LogMiner to extract records. The database must already have after−image journaling enabled.

Log

Nolog

Specifies that the setting of the LogMiner state for the database be reported to SYS$OUTPUT. The default is the setting of the DCL VERIFY flag, which is controlled by the DCL SET VERIFY command.

USAGE NOTES

- To use the RMU Set Logminer command, you must have the RMU$BACKUP, RMU$RESTORE, or RMU$ALTER privilege in the root file access control list (ACL) for the database or the OpenVMS SYSPRV or BYPASS privilege.
- The RMU Set Logminer command requires offline access to the database. The database must be closed and no other users may be accessing the database.
- Execute a full database backup operation after issuing an RMU Set Logminer command that displays the RMU−W−DOFULLBCK warning message. Immediately after enabling LogMiner, you should perform a database after−image journal backup using the RMU Backup After_Journal command.

EXAMPLES

Example 1

The following example enables a database for LogMiner for Rdb operation.

```
$ RMU /SET LOGMINER /ENABLE OLTPDB.RDB
```

7.3 RMU Dump /Header Command Enhanced

The RMU Dump /Header command has been enhanced to indicate the Continuous LogMiner enabled state. If the LogMiner feature is enabled, an additional line output indicates the Continuous LogMiner state as follows:

```
AIJ Journaling...
  – After–image journaling is enabled
        .
        .
        .
  – LogMiner is enabled
    – Continuous LogMiner is enabled
```

7.4 RMU Show Statistics Utility Enhanced

The RMU Show Statistics utility has been enhanced to include a LogMiner Information statistics screen. This screen is available from the Journaling Information menu when the Continuous LogMiner feature is enabled for a database. For each active process running the RMU Unload After_Journal /Continuous command, the state of the process and the last accessed journal block are displayed.

The State information can be one of the following:

Table 7−4 Continuous LogMiner States

State	Description
Inactive	Processing has not yet completed initialization or is shutting down
Hibernating	Waiting for after−image journal write activity
Polling	Sleep/poll state while waiting for after−image journal writing activity; after a short while, if no after−image journal writing occurs, the Continuous LogMiner will enter the Hibernating state
Extracting	Extracting changes from one or more transactions from the after−image journal

Because the AIJ Journal Information screen provides real−time information, the output is not recorded in the binary output file produced using the Output qualifier. Consequently, this screen is not available when you replay a binary file using the Input qualifier.

The RMU Show Statistics utility LogMiner Information screen also displays the current (last known) after−image journal sequence number and end−of−file location on the current node. In a cluster environment, it is possible that these numbers can vary from the actual last written location in the .aij file depending on what node the last writer is running on.

The RMU Show Statistics utility LogMiner Information screen also includes a "Zoom" option to display detailed information about a Continuous LogMiner process.

7.5 AERCP Format

The current format of the AIJ Extract Recovery Control Point (AERCP) is shown in Table 7−5, AERCP Structure.

Table 7−5 AERCP Structure

Field	Length	Content
VERSION_NUMBER	1	Structure Version Number (currently 1)
STRUCTURE_LENGTH	1	Length of Structure (currently 28)
MBZ	2	Must be Zero
MQP_VNO	4	Micro−quiet−point VNO
MQP_VBN	4	Micro−quiet−point VBN
MQP_TSN	8	Micro−quiet−point TSN
LCP_TSN	8	Last extracted TSN

When formatted for text display, the AERCP structure consists of the six fields (the MBZ field is excluded) displayed as unsigned integers separated by dashes as in the following example:

```
1−28−12−7−3202−3202
```

The internal format of the AERCP structure will change in the future.

Chapter 8
Documentation Corrections

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

8.1 Documentation Corrections

8.1.1 RDM$BIND_LOCK_TIMEOUT_INTERVAL Overrides the Database Parameter

Bug 2203700

When starting a transaction, there are three different values that are used to determine the lock timeout interval for that transaction. Those values are:

1. The value specified in the SET TRANSACTION statement
2. The value stored in the database as specified in CREATE or ALTER DATABASE
3. The value of the logical name RDM$BIND_LOCK_TIMEOUT_INTERVAL

The timeout interval for a transaction is the smaller of the value specified in the SET TRANSACTION statement and the value specified in CREATE DATABASE. However, if the logical name RDM$BIND_LOCK_TIMEOUT_INTERVAL is defined, the value of this logical name overrides the value specified in CREATE DATABASE.

The description of how these three values interact, found in several different parts of the Rdb documentation set, is incorrect and will be replaced by the description above.

The lock timeout value in the database can be dynamically modified from the Locking Dashboard in RMU/SHOW STATISTICS. The Per−Process Locking Dashboard can be used to dynamically override the logical name RDM$BIND_LOCK_TIMEOUT_INTERVAL for one or more processes.

8.1.2 New Request Options for RDO, RDBPRE and RDB$INTERPRET

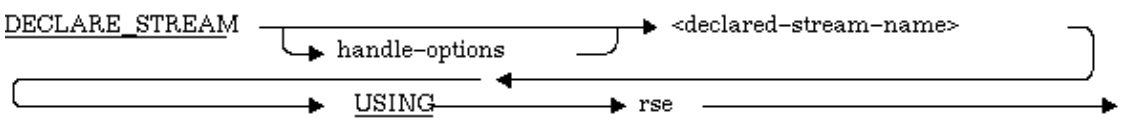
This release note was included in the V70A Release Notes but had gotten dropped somewhere along the line.

For this release of Rdb, two new keywords have been added to the handle−options for the DECLARE_STREAM, the START_STREAM (undeclared format) and FOR loop statements. These changes have been made to RDBPRE, RDO and RDB$INTERPRET at the request of several RDO customers.

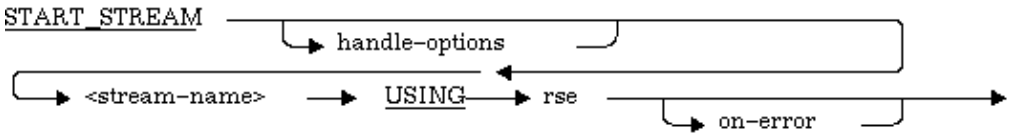
In prior releases, the handle−options could not be specified in interactive RDO or RDB$INTERPRET. This has changed in Rdb but these allowed options will be limited to MODIFY and PROTECTED keywords. For RDBPRE, all options listed will be supported. These option names were chosen to be existing keywords to avoid adding any new keywords to the RDO language.

The altered statements are shown in Example 5−1, Example 5−2 and Example 5−3.

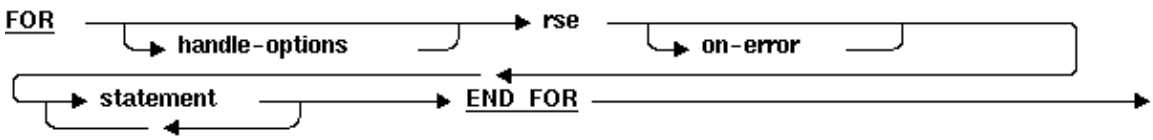
Example 5−1 DECLARE_STREAM Format



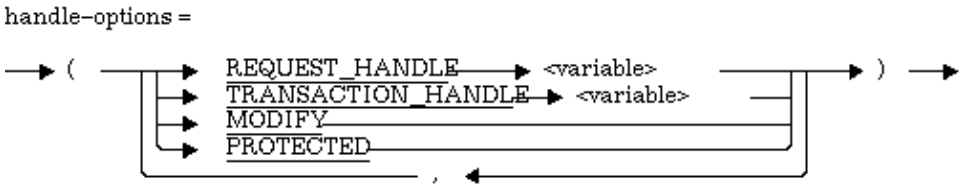
Example 5−2 START_STREAM Format



Example 5−3 FOR Format



Each of these statements references the syntax for the HANDLE−OPTIONS which has been revised and is shown below.



The following options are available for HANDLE−OPTIONS:

- REQUEST_HANDLE specifies the request handle for this request. This option is only valid for RDBPRE and RDML applications. It cannot be used with RDB$INTERPRET, nor interactive RDO.
- TRANSACTION_HANDLE specifies the transaction handle under which this request executes. This option is only valid for RDBPRE and RDML applications. It cannot be used with RDB$INTERPRET, nor interactive RDO.
- MODIFY specifies that the application will modify all (or most) records fetched from the stream or for loop. This option can be used to improve application performance by avoiding lock promotion from SHARED READ for the FETCH to PROTECTED WRITE access for the nested MODIFY or ERASE statement. It can also reduce DEADLOCK occurrence because lock promotions are avoided. This option is valid for RDBPRE, RDB$INTERPRET, and interactive RDO. This option is not currently available for RDML.
  For example:

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

  This FOR loop uses the MODIFY option to indicate that the nested MODIFY is an unconditional statement and so aggressive locking can be undertaken during the fetch of the record in the FOR loop.
- PROTECTED specifies that the application may modify records fetched by this stream by a separate and independent MODIFY statement. Therefore, this stream should be protected from interference (aka Halloween affect). The optimizer will select a snapshot of the rows and store them in a temporary relation for processing, rather than traversing indexes at the time of the FETCH statement. In some cases this may result in poorer performance when the temporary relation is large and overflows from virtual memory to a temporary disk file, but the record stream will be protected from interference. The programmer is directed to the documentation for the Oracle Rdb logical names RDMS$BIND_WORK_VM and RDMS$BIND_WORK_FILE.
  This option is valid for RDBPRE, RDB$INTERPRET, and interactive RDO. This option is not currently available for RDML.

The following example creates a record stream in a BASIC program using Callable RDO:

```
RDMS_STATUS = RDB$INTERPRET ('INVOKE DATABASE PATHNAME "PERSONNEL"')

RDMS_STATUS = RDB$INTERPRET ('START_STREAM (PROTECTED) EMP USING ' + &
                             'E IN EMPLOYEES')

RDMS_STATUS = RDB$INTERPRET ('FETCH EMP')

DML_STRING = 'GET ' +                                          &
                 '!VAL = E.EMPLOYEE_ID;' +                     &
                 '!VAL = E.LAST_NAME;' +                       &
                 '!VAL = E.FIRST_NAME' +                       &
             'END_GET'

RDMS_STATUS = RDB$INTERPRET (DML_STRING, EMP_ID, LAST_NAME, FIRST_NAME)
```

In this case the FETCH needs to be protected against MODIFY statements which execute in other parts of the application.

8.1.3 Missing Descriptions of RDB$FLAGS from HELP File

The HELP file for Oracle Rdb Release 7.0 describes the system tables for Oracle Rdb and was missing these updated descriptions of the RDB$FLAGS column for several tables.

Table 8−1 Changed Columns for RDB$INDICES Table

Column Name	Data Type	Domain Name	Comments
RDB$FLAGS	Integer	RDB$FLAGS	A bit mask where the bits have the following meaning when set:
			Bit 0: This index is of type HASHED.
			Bit 1: This index uses the MAPPING VALUES clause to compress integer value ranges.
			Bit 2: If this is a HASHED index then it is of type ORDERED. If clear this indicates the index is of type SCATTERED.
			Bit 3: Reserved for future use.
			Bit 4: This index has run length compression enabled (ENABLE COMPRESSION).
			Bit 5: This index is no longer used (MAINTENANCE IS DISABLED).
			Bit 6 through 10: Reserved for future use.
			Bit 11: This index has duplicates compressed (DUPLICATES ARE COMPRESSED).
			Bit 12: This index is of type SORTED RANKED.
			Bits 13 through 31: Reserved for future use.

Table 8−2 Changed Columns for RDB$RELATIONS Table

Column Name	Data Type	Domain Name	Comments

RDB$FLAGS	Integer	RDB$FLAGS	A bit mask where the bits have the following meaning when set:
			Bit 0: This relation is a view.
			Bit 1: This relation is not compressed.
			Bit 2: The SQL clause, WITH CHECK OPTION, is used in this view definition.
			Bit 3: Indicates a special internal system relation.
			Bit 4: This view is not an ANSI updatable view.
			Bit 5: This is an imported table in the Distributed Option for Rdb catalog.
			Bit 6: This is a passthru table in the Distributed Option for Rdb catalog.
			Bit 7: This is a partitioned view in the Distributed Option for Rdb catalog.
			Bit 8: This table has compression defined by the storage map. When set Bit 1 in this bit mask is ignored.
			Bit 9: This is a temporary table.
			Bit 10: When bit 9 is set this is a global temporary table, when clear it indicates a local temporary table.
			Bit 11: When bit 9 is set this indicates that the rows in the temporary table should be deleted upon COMMIT.
			Bit 12: Reserved for future use.
			Bit 13: A table (via a computed by column) or view references a local temporary table.
			Bit 14: Reserved for future use.
			Bit 15: This is a system table with a special storage map.
			Bits 16 through 31: Reserved for future use.

Table 8−3 Changed Columns for RDB$STORAGE_MAPS Table

Column Name	Data Type	Domain Name	Comments
RDB$FLAGS	Integer	RDB$FLAGS	A bit mask where the bits have the following meaning when set:
			Bit 0: This table or index is mapped to page format MIXED areas.
			Bit 1: This partition is not compressed.
			Bit 2: This is a strictly partitioned storage map, the partitioning columns become read only for UPDATE.
			Bit 3 through 31: Reserved for future use.

8.1.4 A Way to Find the Transaction Type of a Particular Transaction Within the Trace Database

The table EPC$1_221_TRANSACTION in the formatted Oracle Trace database has a column LOCK_MODE_START of longword datatype. The values of this column indicate the type of transaction a particular transaction was.

```
Value           Transaction type
-----           ----------------
8               Read only
```

```
9               Read write
14              Batch update
```

8.1.5 Clarification of SET FLAGS Option DATABASE_PARAMETERS

Bug 1668049

The Oracle Rdb7 SQL Reference Manual describes the option DATABASE_PARAMETERS in Table 7−6 in the SET FLAGS section. However, this keyword generates output only during ATTACH to the database which happens prior to the SET FLAGS statement executing.

This option is therefore only useful when used with the RDMS$SET_FLAGS logical name which provides similar functionality.

```
$ define RDMS$SET_FLAGS "database_parameters"
$ sql$
SQL> Attach 'File db$:scratch';
 ATTACH #1, Database BLUGUM$DKA300:[SMITHI.DATABASES.V70]SCRATCH.RDB;1
~P Database Parameter Buffer (version=2, len=79)
0000 (00000) RDB$K_DPB_VERSION2
0001 (00001) RDB$K_FACILITY_ALL
0002 (00002) RDB$K_DPB2_IMAGE_NAME "NODE::DISK:[DIR]SQL$70.EXE;1"
0040 (00064) RDB$K_FACILITY_ALL
0041 (00065) RDB$K_DPB2_DBKEY_SCOPE (Transaction)
0045 (00069) RDB$K_FACILITY_ALL
0046 (00070) RDB$K_DPB2_REQUEST_SCOPE (Attach)
004A (00074) RDB$K_FACILITY_RDB_VMS
004B (00075) RDB$K_DPB2_CDD_MAINTAINED (No)
 RDMS$BIND_WORK_FILE = "DISK:[DIR]RDMSTTBL$UEOU3LQ0RV2.TMP;" (Visible = 0)
SQL> Exit
 DETACH #1
```

8.1.6 Additional Information About Detached Processes

Oracle Rdb documentation omits necessary detail on running Oracle Rdb from a detached process.

Applications run from a detached process must ensure that the OpenVMS environment is established correctly before running Oracle Rdb. Otherwise, Oracle Rdb will not execute.

Attempts to attach to a database and execute an Oracle Rdb query from applications running as detached processes will result in an error similar to the following:

```
%RDB−F−SYS_REQUEST, error from system services request
−SORT−E−OPENOUT, error opening {file} as output
−RMS−F−DEV, error in device name or inappropriate device type for
operation
```

The problem occurs because a detached process does not normally have the logical names SYS$LOGIN or SYS$SCRATCH defined.

There are two methods that can be used to correct this:

1. Use the DCL command procedure RUN_PROCEDURE to run the ACCOUNTS application: RUN_PROCEDURE.COM includes the single line:

```
$ RUN ACCOUNTS_REPORT
```

Then execute this procedure using this command:

```
$ RUN/DETACH/AUTHORIZE SYS$SYSTEM:LOGINOUT/INPUT=RUN_PROCEDURE
```

This solution executes SYS$SYSTEM:LOGINOUT so that the command language interface (DCL) is activated. This causes the logical names SYS$LOGIN and SYS$SCRATCH to be defined for the detached process. The /AUTHORIZE qualifier also ensures that the users' process quota limits (PQLs) are used from the system authorization file rather than relying on the default PQL system parameters, which are often insufficient to run Oracle Rdb.

2. If DCL is not desired, and SYS$LOGIN and SYS$SCRATCH are not defined, then prior to executing any Oracle Rdb statement, you should define the following logical names:
   - RDMS$BIND_WORK_FILE
     Define this logical name to allow you to reduce the overhead of disk I/O operations for matching operations when used in conjunction with the RDMS$BIND_WORK_VM logical name. If the virtual memory file is too small, then overflow to disk will occur at the disk and directory location specified by RDMS$BIND_WORK_FILE.
     For more information on RDMS$BIND_WORK_FILE and RDMS$BIND_WORK_VM, see the Oracle Rdb Guide to Database Performance and Tuning.
   - SORTWORK0, SORTWORK1, and so on
     The OpenVMS sort/merge utility (SORT/MERGE) attempts to create sort work files in SYS$SCRATCH. If the SORTWORK logical names exist, the utility will not require the SYS$SCRATCH logical. However, note that not all queries will require sorting, and that some sorts will be completed in memory and so will not necessarily require disk space.
     If you use the logical RDMS$BIND_SORT_WORKFILES, you will need to define further SORTWORK logical names as described in the Oracle Rdb Guide to Database Performance and Tuning.

You should also verify that sufficient process quotas are specified on the RUN/DETACH command line, or defined as system PQL parameters to allow Oracle Rdb to execute.

8.1.7 The Halloween Problem

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

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

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

Oracle Rdb avoids this problem if it knows that the cursors subject table will be updated. For example, if the SQL syntax UPDATE ... WHERE CURRENT OF is used to perform updates of target rows, or if the RDO/RDML MODIFY statement uses the context variable for the stream. Then the optimizer will choose an alternate access strategy if an update can occur which may cause the

Halloween problem. This can be seen in the access strategy in the example below as a "Temporary relation" being created to hold the result of the cursor query.

When you use interactive or dynamic SQL, the UPDATE ... WHERE CURRENT OF or DELETE ... WHERE CURRENT OF statements will not be seen until after the cursor is declared and opened. In these environments, you must use the FOR UPDATE clause to specify that columns selected by the cursor will be updated during cursor processing. This is an indication to the Rdb optimizer so that it protects against the Halloween problem in this case. This is shown in the two examples below Example 8−1 and Example 8−2.

Example 8−1 shows that the EMP_LAST_NAME index is used for retrieval. Any update performed will possibly be subject to the Halloween problem.

Example 8−1 Interactive Cursor with no Halloween Protection

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

Example 8−2 shows that the query specifies that the column LAST_NAME will be updated by some later query. Now the optimizer protects the EMP_LAST_NAME index used for retrieval by using a "Temporary Relation" to hold the query result set. Any update performed on LAST_NAME will now avoid the Halloween problem.

Example 8−2 Interactive Cursor with Halloween Protection

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

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

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

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

8.1.8 RDM$BIND_MAX_DBR_COUNT Documentation Clarification

Bug 1495227

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

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

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

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

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

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

8.1.9 RMU /UNLOAD /AFTER_JOURNAL NULL Bit Vector Clarification

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

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

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

```
/* DATATYPES.C */

#include <stdio.h>
#include <descrip.h>
```

```
#include <starlet.h>
#include <string.h>

#pragma member_alignment __save
#pragma nomember_alignment

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

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

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

#pragma member_alignment __restore

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


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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
SQL> ATTACH 'FILE MF_PERSONNEL';
SQL> CREATE TABLE DATATYPES (
     F_TINYINT TINYINT
    ,F_SMALLINT SMALLINT
    ,F_INTEGER INTEGER
    ,F_BIGINT BIGINT
    ,F_DOUBLE DOUBLE PRECISION
    ,F_REAL REAL
    ,F_FIXSTR CHAR (10)
```

```
    ,F_VARSTR VARCHAR (10));
SQL> COMMIT;
SQL> INSERT INTO DATATYPES VALUES (1, NULL, 2, NULL, 3, NULL, 'THIS', NULL);
SQL> INSERT INTO DATATYPES VALUES (NULL, 4, NULL, 5, NULL, 6, NULL, 'THAT');
SQL> COMMIT;
SQL> EXIT;
$ RMU /BACKUP /AFTER_JOURNAL MF_PERSONNEL AIJBCK.AIJ
$ RMU /UNLOAD /AFTER_JOURNAL MF_PERSONNEL AIJBCK.AIJ -
    /TABLE = (NAME=DATATYPES, OUTPUT=DATATYPES.DAT)
$ CC DATATYPES.C
$ LINK DATATYPES.OBJ
$ RUN DATATYPES.EXE
```

8.1.10 Location of Host Source File Generated by the SQL Precompilers

Bug 478898

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

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

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

Directory MYDISK:[LUND]

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

Total of 3 files.
```

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

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

Directory MYDISK:[LUND]

SCC_TRY_MLI_SUCCESSFUL.SC;2

Total of 1 file.
LUND> dir myobj.*

Directory MYDISK:[LUND]

MYOBJ.C;1           MYOBJ.OBJ;2

Total of 2 files.

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

```
LUND> dir scc_try_mli_successful.*

Directory MYDISK:[LUND]

SCC_TRY_MLI_SUCCESSFUL.SC;2

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

Directory MYDISK:[LUND.TMP]

SCC_TRY_MLI_SUCCESSFUL.C;1              SCC_TRY_MLI_SUCCESSFUL.OBJ;2

Total of 2 files.
```

This problem has been corrected in Oracle Rdb Release 7.0.6.

8.1.11 Suggestion to Increase GH_RSRVPGCNT Removed

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

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

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

8.1.12 Clarification of the DDLDONOTMIX Error Message

Bug 454080

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

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

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

```
some ALTER DATABASE clauses
```

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

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

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

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

8.1.13 Compressed Sorted Index Entry Stored in Incorrect Storage Area

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

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

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

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

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

The following example shows the problem:

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

create table T1
        (C1 integer);
```

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

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

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

Note

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

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

The optimized algorithm now only scans the relevant index areas (and no longer skips over empty areas) resulting in only those rows being returned. Therefore, it is recommended that the index be dropped and re−created. For a short term solution, another alternative is to disable the new optimization by defining the logical RDMS$INDEX_PART_CHECK to 0.

This problem has been corrected in Oracle Rdb Release 7.0.1.3.

8.1.14 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*.

8.1.15 Oracle Rdb Logical Names

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

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

8.1.16 Waiting for Client Lock Message

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

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

The "waiting for client lock" message means the database user is requesting an incompatible locking mode. For example, when trying to delete a table which is in use, the drop operation requests a

PROTECTED WRITE lock on the metadata object (such as a table) which is incompatible with the existing PROTECTED READ lock currently used by others of the table.

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

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

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

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

Table 8–4 Object Type Values

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

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

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

```
Process.ID Since......   Stall.reason............................. Lock.ID.
46001105:2 10:40:46.38 – waiting for client '........' 000000190000000400000055
                                                      (1)        (2)       (3)      (4)
```

The following list describes each part of the client lock:

1. ..... indicates nonprintable characters.
2. 00000019 indicates unique identifier hex value 19 (RDB$RELATION_ID = 25).
3. 00000004 indicates object type 4 which is a table.
4. 00000055 indicates this is a client lock.

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

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

Note

Because the full client lock output is long, it may require more space than is allotted for the Stall.reason column and therefore can be overwritten by the Lock.ID. column output.

For more detailed lock information, perform the following steps:

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

8.1.17 Documentation Error in *Oracle Rdb Guide to Database Performance and Tuning*

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

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

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

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

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

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

8.1.18 SET FLAGS Option IGNORE_OUTLINE Not Available

Bug 510968

The *Oracle Rdb7 SQL Reference Manual* described the option IGNORE_OUTLINE in Table 7−6 of the SET FLAGS section. However, this keyword was not implemented in Oracle Rdb Release 7.0.

This has been corrected in this release of Oracle Rdb. 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.

8.1.19 SET FLAGS Option INTERNALS Not Described

The *Oracle Rdb7 SQL Reference Manual* does not describe the option INTERNALS in Table 7−6 in the SET FLAGS section. This keyword was available in first release of Oracle Rdb 7.0 and is used to enable debug flags output for internal queries such as constraints and triggers. It can be used in conjunction with other options such as STRATEGY, BLR, and EXECUTION. For example, the following flag settings are equivalent to defining the RDMS$DEBUG_FLAGS as ISn and shows the strategy used by the trigger's actions on the AFTER DELETE trigger on the EMPLOYEES table.

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

Alias RDB$DBHANDLE:
```

```
Flags currently set for Oracle Rdb:
   INTERNALS,STRATEGY,PREFIX,REQUEST_NAMES
SQL> DELETE FROM EMPLOYEES WHERE EMPLOYEE_ID = '00164';
~S: Trigger name  EMPLOYEE_ID_CASCADE_DELETE
Get     Temporary relation      Retrieval by index of relation DEGREES
  Index name  DEG_EMP_ID [1:1]
~S: Trigger name  EMPLOYEE_ID_CASCADE_DELETE
Get     Temporary relation      Retrieval by index of relation JOB_HISTORY
  Index name  JOB_HISTORY_HASH [1:1]
~S: Trigger name  EMPLOYEE_ID_CASCADE_DELETE
Get     Temporary relation      Retrieval by index of relation SALARY_HISTORY
  Index name  SH_EMPLOYEE_ID [1:1]
~S: Trigger name  EMPLOYEE_ID_CASCADE_DELETE
Conjunct        Get     Retrieval by index of relation DEPARTMENTS
  Index name  DEPARTMENTS_INDEX [0:0]
Temporary relation      Get     Retrieval by index of relation EMPLOYEES
  Index name  EMPLOYEES_HASH [1:1]       Direct lookup
1 row deleted
```

8.1.20 Documentation for VALIDATE_ROUTINE Keyword for SET FLAGS

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

This example shows the re−validation of a stored procedure. When the stored routine is successfully prepared (but not executed), the setting of VALIDATE_ROUTINE causes the entry for this routine in the RDB$ROUTINES system table to be set as valid.

```
SQL> SET TRANSACTION READ WRITE;
SQL> SET FLAGS 'VALIDATE_ROUTINE';
SQL> SET NOEXECUTE;
SQL> CALL ADD_EMPLOYEE ('Smith');
SQL> SET EXECUTE;
SQL> COMMIT;
```

In this example, the use of the SET NOEXECUTE statement in interactive SQL allows the stored routine to be successfully compiled, but it is not executed.

8.1.21 Documentation for Defining the RDBSERVER Logical Name

Bugs 460611 and 563649.

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

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

These definitions are inconsistent with other command procedures that attempt to reference the RDBSERVERxx.EXE image. The following is one example where the RDBSERVER.COM

procedure references SYS$COMMON:<SYSEXE> and SYS$COMMON:[SYSEXE], rather than SYS$SYSTEM:

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

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

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

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

8.1.22 Undocumented SET Commands and Language Options

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

8.1.22.1 QUIET COMMIT Option

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

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

Note

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

Examples

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

```
SQL> COMMIT;
%SQL-F-NO_TXNOUT, No transaction outstanding
```

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

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

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

PROCEDURE S_TXN (SQLCODE);
SET TRANSACTION READ WRITE;

PROCEDURE C_TXN (SQLCODE);
COMMIT;
```

8.1.22.2 COMPOUND TRANSACTIONS Option

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

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

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

Examples

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

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

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

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

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

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

8.1.23 Undocumented Size Limit for Indexes with Keys Using Collating Sequences

Bug 586079

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

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

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

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

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

8.1.24 Changes to RMU/REPLICATE AFTER/BUFFERS Command

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

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

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

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

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

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

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

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

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

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

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

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

8.1.26 CREATE INDEX Supported for Hot Standby

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

8.1.27 Dynamic OR Optimization Formats

Bug 711643

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

Chapter 9
Known Problems and Restrictions

This chapter describes problems, restrictions, and workarounds known to exist in Oracle Rdb Release 7.0.6.4.

9.0.1 Optimization of Check Constraints

Bug 1448422

When phrasing constraints using the "CHECK" syntax, a poorer strategy can be chosen by the optimizer than when the same or similar constraint is phrased using referential integrity (PRIMARY and FOREIGN KEY) constraints. Following is an example.

I have two tables T1 and T2, both with one column, and I wish to ensure that all values in table T1 exist in T2. Both tables have an index on the referenced field. I could use a PRIMARY KEY constraint on T2 and a FOREIGN KEY constraint on T1.

```
SQL> alter table t2
cont>   alter column f2 primary key not deferrable;
SQL> alter table t1
cont>   alter column f1 references t2 not deferrable;
```

When deleting from the PRIMARY KEY table, Oracle Rdb will only check for rows in the FOREIGN KEY table where the FOREIGN KEY has the deleted value. This can be seen as an index lookup on T1 in the retrieval strategy.

```
SQL> delete from t2 where f2=1;
Get     Temporary relation      Retrieval by index of relation T2
  Index name  I2 [1:1]
Index only retrieval of relation T1
  Index name  I1 [1:1]
%RDB-E-INTEG_FAIL, violation of constraint T1_FOREIGN1 caused operation to fail
```

The failure of the constraint is not important. What is important is that Rdb efficiently detects that only those rows in T1 with the same values as the deleted row in T2 can be affected.

It is necessary sometimes to define this type of relationship using CHECK constraints. This could be necessary because the presence of NULL values in the table T2 precludes the definition of a primary key on that table. This could be done with a CHECK constraint of the form:

```
SQL> alter table t1
cont>   alter column f1
cont>   check (f1 in (select * from t2)) not deferrable;
SQL> delete from t2 where f2=1;
Get     Temporary relation      Retrieval by index of relation T2
  Index name  I2 [1:1]
Cross block of 2 entries
  Cross block entry 1
    Index only retrieval of relation T1
      Index name  I1 [0:0]
  Cross block entry 2
    Conjunct        Aggregate-F1    Conjunct
    Index only retrieval of relation T2
      Index name  I2 [0:0]
%RDB-E-INTEG_FAIL, violation of constraint T1_CHECK1 caused operation to fail
```

The cross block is for the constraint evaluation. This retrieval strategy indicates that to evaluate the constraint, the entire index on table T1 is being scanned and for each key, the entire index in table T2 is being scanned.

The behavior can be improved somewhat by using an equality join condition in the select clause of the constraint:

```
SQL> alter table t1
cont>   alter column f1
cont>   check (f1 in (select * from t2 where f2=f1))
cont>      not deferrable;

or:

SQL> alter table t1
cont>   alter column f1
cont>   check (f1=(select * from t2 where f2=f1))
cont>      not deferrable;
```

In both cases, the retrieval strategy will look as follows:

```
SQL> delete from t2 where f2=1;
Get     Temporary relation      Retrieval by index of relation T2
  Index name  I2 [1:1]
Cross block of 2 entries
  Cross block entry 1
    Index only retrieval of relation T1
      Index name  I1 [0:0]
  Cross block entry 2
    Conjunct        Aggregate-F1    Conjunct
    Index only retrieval of relation T2
      Index name  I2 [1:1]
%RDB-E-INTEG_FAIL, violation of constraint T1_CHECK1 caused operation to fail
```

While the entire T1 index is scanned, at least the value from T1 is used to perform an index lookup on T2.

These restrictions result from semantic differences in the behavior of the "IN" and "EXISTS" operators with respect to null handling, and the complexity of dealing with non-equality join conditions.

To improve the performance of this type of integrity check on larger tables, it is possible to use a series of triggers to perform the constraint check. The following triggers perform a similar check to the constraints above.

```
SQL> create trigger t1_insert
cont>  after insert on t1
cont>  when (not exists (select * from t2 where f2=f1))
cont>    (error) for each row;
SQL> create trigger t1_update
cont>  after update on t1
cont>  when (not exists (select * from t2 where f2=f1))
cont>    (error) for each row;
SQL> ! A delete trigger is not needed on T1.
SQL> create trigger t2_delete
cont>  before delete on t2
cont>  when (exists (select * from t1 where f1=f2))
cont>    (error) for each row;
SQL> create trigger t2_modify
cont>  after update on t2
cont>  referencing old as t2o new as t2n
```

```
cont>  when (exists (select * from t1 where f1=t2o.f2))
cont>    (error) for each row;
SQL> ! An insert trigger is not needed on T2.
```

The strategy for a delete on T2 is now:

```
SQL> delete from t2 where f2=1;
Aggregate-F1    Index only retrieval of relation T1
  Index name  I1 [1:1]
Temporary relation      Get     Retrieval by index of relation T2
  Index name  I2 [1:1]
%RDB-E-TRIG_INV_UPD, invalid update; encountered error condition defined for
trigger
-RDMS-E-TRIG_ERROR, trigger T2_DELETE forced an error
```

The trigger strategy is the index only retrieval displayed first. You will note that the index on T1 is used to examine only those rows that may be affected by the delete.

Care must be taken when using this workaround as there are semantic differences in the operation of the triggers, the use of "IN" and "EXISTS", and the use of referential integrity constraints.

This workaround is useful where the form of the constraint is more complex and cannot be phrased using referential integrity constraints. For example, if the application is such that the value in table T1 may be spaces or NULLs to indicate the absence of a value, the above triggers could easily be modified to allow for these semantics.

9.0.2 Dynamic Optimization Estimation Incorrect for Ranked Indices

The dynamic optimization process was incorrectly calculating the cost of scanning indices of type *SORTED RANKED*.

In the following example, the table being queried has the numbers one to one thousand in both fields. The different ranges used should result in a different estimated cost. However, in all cases the *ESTIM* phase computes the cost of scanning these indicies as 680:

```
SQL> select * from t where f1 between 1 and 2 and f2 between 2 and 1000;
~S#0003
Leaf#01 FFirst T Card=1000
  BgrNdx1 T1 [1:1] Fan=17
  BgrNdx2 T2 [1:1] Fan=17
~E#0003.01(1) Estim   Ndx:Lev/Seps/DBKeys 1:34/0\680 2:34/0\680
~E#0003.01(1) BgrNdx1 EofData  DBKeys=2  Fetches=2+0  RecsOut=1 #Bufs=1
~E#0003.01(1) FgrNdx  FFirst   DBKeys=1  Fetches=0+1  RecsOut=1`ABA
~E#0003.01(1) Fin     Buf      DBKeys=2  Fetches=0+0  RecsOut=1
          F1            F2
           2             2
1 row selected
SQL> select * from t where f1 between 2 and 1000 and f2 between 1 and 2;
~S#0004
Leaf#01 FFirst T Card=1000
  BgrNdx1 T1 [1:1] Fan=17
  BgrNdx2 T2 [1:1] Fan=17
~E#0004.01(1) Estim   Ndx:Lev/Seps/DBKeys 1:34/0\680 2:34/0\680
~E#0004.01(1) BgrNdx1 EofData  DBKeys=999  Fetches=0+10  RecsOut=1 #Bufs=10
~E#0004.01(1) FgrNdx  FFirst   DBKeys=1  Fetches=0+11  RecsOut=1`ABA
~E#0004.01(1) Fin     Buf      DBKeys=999  Fetches=0+0  RecsOut=1
          F1            F2
           2             2
```

```
1 row selected
```

In the first example (query 3), the index T1 on field F1 is the correct index to use, as the key range is very small. In the second example (query 4), the index T2 on field F2 is the correct index to use. However, in both cases, the indices are costed the same so no index reordering takes place.

Even in this small example, significantly more work is being performed in query 4 as can be observed from the I/O counts.

This is a known problem in Oracle Rdb and it will be fixed in a future release.

The only workaround is to use indices of *TYPE IS SORTED* rather than of *TYPE IS SORTED RANKED*.

9.0.3 Running Rdb Applications With the VMS Heap Analyzer

When trying to debug an Rdb application under the OpenVMS Heap Analyzer (by defining LIBRTL as SYS$LIBRARY:LIBRTL_INSTRUMENTED), the software will not attach to the database, and returns

```
     RDB-E-UNAVAILABLE, Oracle Rdb is not available on your system
```

as if RDB is not running.

To solve this problem, there are two executables that must be installed as known images:

```
$install add sys$share:librtl_instrumented
$install add sys$share:dgit$libshr12
```

The error is misleading. Since parts of Rdb are installed as privileged images, any shareable images it references, AND any images they, in turn, reference, must also be 'known'. By redirecting LIBRTL to SYS$LIBRARY:LIBRTL_INSTRUMENTED, these extra images are referenced. If Rdb had directly referenced the new image, a more accurate error, such as:

```
%DCL-W-ACTIMAGE, error activating image xxxxx
```

would have been reported.

9.0.4 RMU/RECOVER/AREA Needs Area List

Bug 1778243

When doing an RMU/RECOVER/AREA, without specifying a list of area names, there will be a new version of the current active AIJ file created. This new version of the AIJ will have the next recovery sequence number. If a subsequent recovery is applied, an error is generated indicating that the original recovery sequence number cannot be found and the recovery will abort.

If a list of storage areas to be recovered is supplied, this behaviour does not occur and no new version of the journal is created. It is recommended as best practice to use a list of storage areas when recovering by area to avoid any subsequent confusion during recovery.

9.0.5 PAGE TRANSFER VIA MEMORY Disabled

Oracle internal testing has revealed that the "PAGE TRANSFER VIA MEMORY" option for global buffers is not as robust as is needed for the Mission Critical environments where Oracle Rdb is often deployed. This feature has been disabled in Oracle Rdb Version 7.0.xx. Oracle intends to re−enable this feature in a future Version 7.1 release.

9.0.6 RMU/VERIFY Reports PGSPAMENT or PGSPMCLST Errors

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

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

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

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

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

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

- Recreate the database by performing:
    1. SQL EXPORT
    2. SQL DROP DATABASE
    3. SQL IMPORT
- Recreate the database by performing:
    1. RMU/BACKUP
    2. SQL DROP DATABASE

3. RMU/RESTORE

- Repair the SPAM pages by using the RMU/REPAIR command. Note that the RMU/REPAIR command does not write its changes to an after−image journal (AIJ) file. Therefore, Oracle recommends that a full database backup be performed immediately after using the RMU/REPAIR command.

9.0.7 Behavior Change in 'With System Logical_Name Translation' Clause

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

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

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

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

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

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

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

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

9.0.8 Carry−Over Locks and NOWAIT Transactions Clarification

In NOWAIT transactions, the BLAST mechanism cannot be used. For the blocking user to receive the BLAST signal, the requesting user must request the locked resource with WAIT (which a NOWAIT transaction does not do). Oracle Rdb defines a resource called NOWAIT, which is used to indicate that a NOWAIT transaction has been started. When a NOWAIT transaction starts, the user requests the NOWAIT resource. All other database users hold a lock on the NOWAIT resource so that when the NOWAIT transaction starts, all other users are notified with a NOWAIT BLAST. The BLAST causes blocking users to release any carry−over locks. There can be a delay before the transactions with carry−over locks detect the presence of the NOWAIT transaction and release their carry−over locks. You can detect this condition by

examining the stall messages. If the "Waiting for NOWAIT signal (CW)" stall message appears frequently, then the application is probably experiencing a decrease in performance and you should consider disabling the carry−over lock behavior.

9.0.9 Strict Partitioning May Scan Extra Partitions

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

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

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

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

9.0.10 Exclusive Access Transactions May Deadlock With RCS Process

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

There are at least three suggested workarounds to this problem:

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

9.0.11 Oracle Rdb and OpenVMS ODS−5 Volumes

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

- A new, optional, volume structure, ODS−5, which provides support for file names that are longer and

have a greater range of legal characters than in previous versions of OpenVMS.
- Support for "deep" directory trees.

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

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

Oracle does support Oracle Rdb database file components on ODS−5 volumes provided that all of these files and directories used by Oracle Rdb strictly follow the ODS−2 file and directory name conventions. In particular, all file names must be specified entirely in uppercase and "special" characters in file or directory names are forbidden.

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

Bug 551240

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

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

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

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

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

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

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

Bug 413410

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

Consider this example:

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

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

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

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

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

Bug 755182

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

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

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

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

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

Bug 1094071

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

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

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

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

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

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

9.0.16 Application and Oracle Rdb Both Using SYS$HIBER

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

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

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

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

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

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

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

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

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

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

9.0.17 IMPORT Unable to Import Some View Definitions

Bug 520651

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

The following example shows the errors from IMPORT.

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

The following script can be used to demonstrate the problem.

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

create module TFORMAT
    language SQL

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

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

commit;

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

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

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

9.0.18 AIJSERVER Privileges

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

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

- ALTPRI
  This privilege allows the AIJSERVER to adjust its own priority to ensure adequate quorum (CPU utilization) to prompt message processing.
- PSWAPM
  This privilege allows the AIJSERVER to enable and disable process swapping, also necessary to ensure prompt message processing.
- SETPRV
  This privilege allows the AIJSERVER to temporarily set any additional privileges it may need to access the standby database or its server processes.
- SYSPRV
  This privilege allows the AIJSERVER to access the standby database rootfile, if necessary.
- WORLD
  This privilege allows the AIJSERVER to more accurately detect standby database server process failure and handle network failure more reliably.

9.0.19 Lock Remastering and Hot Standby

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

- Disable dynamic lock remastering. This can be done dynamically by setting the SYSGEN parameter PE1 to the value 1.
  When using this option, be sure that Hot Standby is started on the node where the standby database is first opened.
- Increasing the LOCKDIRWT value for the LRS node higher than any other node in the same cluster. However, this is not a dynamic SYSGEN parameter, and a node re−boot is required.

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

9.0.20 RDB_SETUP Privilege Error

Rdb Web Agent V3.0 exposes a privilege problem with Rdb V7.0 and later. This will be fixed in the next Rdb 7.0 release.

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

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

9.0.21 Starting Hot Standby on Restored Standby Database May Corrupt Database

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

WORKAROUND: None.

9.0.22 Restriction on Compound Statement Nesting Levels

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

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

BEGIN
  DECLARE ....;

SET :VARS = 0;

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

```
                        ELSE
                          IF :FLAG < 0 THEN SET...;      ! #7
                          END IF;                        ! #7
                        END IF;                         ! #6
                      END IF;                           ! #5

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

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

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

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

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

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

9.0.24 Concurrent DDL and Read−Only Transaction on the Same Table Not Compatible

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

The following example shows the actions of the two transactions:

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

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

9.0.25 Oracle Rdb and the SRM_CHECK Tool

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

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

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

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

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

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

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

9.0.26 Oracle RMU Checksum_Verification Qualifier

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

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

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

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

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

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

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

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

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

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

9.0.28 Restriction on Using /NOONLINE with Hot Standby

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

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

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

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

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

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

9.0.29 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 Rdb. This corruption has been fixed and indexes created using Release 7.0.1.3 will not be impacted.

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

9.0.30 DBAPack for Windows 3.1 is Deprecated

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

9.0.31 Determining Mode for SQL Non−Stored Procedures

Bug 506464.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
end;
```

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

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

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

9.0.32 DROP TABLE CASCADE Results in %RDB−E−NO_META_UPDATE Error

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

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

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

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

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

The workaround to this problem is to first delete the storage map, and then delete the table using the

CASCADE option. The following example shows the workaround. The SHOW statement indicates that the table, index, and storage map were deleted:

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

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

9.0.33 Bugcheck Dump Files with Exceptions at COSI_CHF_SIGNAL

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

For example, consider the following bugcheck file stack information:

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

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

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

9.0.34 Interruptions Possible when Using Multistatement or Stored Procedures

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

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

Session 1

```
SQL> ATTACH 'FILE MF_PERSONNEL';
SQL> CREATE FUNCTION LIB$WAIT (IN REAL BY REFERENCE)
```

```
cont> RETURNS INT;
cont> EXTERNAL NAME LIB$WAIT
cont> LOCATION 'SYS$SHARE:LIBRTL.EXE'
cont> LANGUAGE GENERAL
cont> GENERAL PARAMETER STYLE
cont> VARIANT;
SQL> COMMIT;
SQL> EXIT;

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

Session 2

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

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

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

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

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

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

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

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

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

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

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

9.0.36 Hot Standby Replication Waits when Starting if Read−Only Transactions Running

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

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

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

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

The following example shows the error:

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

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

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

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

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

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

9.0.38 DEC C and Use of the /STANDARD Switch

Bug 394451

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

9.0.40 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.

9.0.41 Restriction Using Backup Files Created Later than Oracle Rdb Release 7.0.1

Bug 521583

Backup files created using Oracle Rdb releases later than 7.0.1 cannot be restored using Oracle Rdb 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 Rdb 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 Rdb.

9.0.42 RMU Backup Operations and Tape Drive Types

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

Oracle RMU attempts to prevent using different tape drive densities during a backup operation, but is not able to detect all invalid cases and expects that all tape drives for a backup are of the same type.

As long as all the tapes used during a backup operation can be read by the same type of tape drive during a restore operation, the backup is likely to be valid. This may be the case, for example, when using a TA90 and a TA90E.

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

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

9.0.43 Use of Oracle Rdb from Shared Images

Bug 470946

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

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

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

9.0.44 Restriction Added for CREATE STORAGE MAP on Table with Data

Oracle Rdb Releease 7.0 added support that allows a storage map to be added to an existing table which contains data. The restrictions listed for Oracle Rdb Release 7.0 were:

- The storage map must be a simple map that references only the default storage area and represents the current (default) mapping for the table. The default storage area is either RDB$SYSTEM or the area

name provided by the CREATE DATABASE...DEFAULT STORAGE AREA clause.
- The new map cannot change THRESHOLDS or COMPRESSION for the table, nor can it use the PLACEMENT VIA INDEX clause. It can only contain one area and cannot be vertically partitioned. This new map simply describes the mapping as it exists by default for the table.

This release of Rdb 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
```

9.0.45 Oracle Rdb Workload Collection Can Stop Hot Standby Replication

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

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

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

Workaround

To work around this problem, perform the following:

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

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

  This change is propagated to the standby database automatically when you restore the standby database and restart replication operations. Note that, by default, the workload collection feature is disabled. You need to disable workload collection only if you previously enabled workload collection with the WORKLOAD COLLECTION IS ENABLED clause.
- On the standby database, include the Transaction_Mode qualifier on the RMU/Restore command when you restore the standby database. You should set this qualifier to read−only to prevent modifications to the standby database when replication operations are not active. The following example shows the Transaction_Mode qualifier used in a typical RMU/Restore command:

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

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

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

9.0.46 RMU Convert Command and System Tables

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

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

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

9.0.47 Converting Single−File Databases

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

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

9.0.48 Restriction when Adding Storage Areas with Users Attached to Database

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

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

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

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

9.0.49 Restriction on Tape Usage for Digital UNIX V3.2

9.0.50 Support for Single−File Databases to be Dropped in a Future Release

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

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

- Use the Oracle RMU commands, such as Backup and Restore, to make copies, back up, or move single−file databases. Do not use operating system commands to copy, back up, or move databases.
- Create new databases as multifile databases even though single−file databases are supported in Oracle Rdb release 6.1 and release 7.0.

9.0.51 DECdtm Log Stalls

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

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

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

```
$ MCR LMCP SET NOTIMEZONE
```

This should force the log to restart.

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

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

This command displays output similar to the following:

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

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

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

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

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

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

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

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

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

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

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

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

9.0.53 Multiblock Page Writes May Require Restore Operation

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

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

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

9.0.54 Oracle Rdb Network Link Failure Does Not Allow DISCONNECT to Clean Up Transactions

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

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

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

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

When a copy process completes updates to its respective remote database, it updates the RDB$TRANSFERS system table and then tries to delete any RDB$CHANGES rows not needed by any transfers. During this process, the RDB$CHANGES table cannot be updated by any application process, holding up any database updates until the deletion process is complete. The application stalls while waiting for the RDB$CHANGES table. The resulting contention for RDB$CHANGES SPAM pages and data pages severely impacts performance throughput, requiring user intervention with normal processing.

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

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

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

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

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

The following example demonstrates this restriction:

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

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

SQL>
```

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

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

9.0.57 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.)

9.0.58 Different Methods of Limiting Returned Rows from Queries

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

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

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

  Interactive SQL must load its metadata cache for the table before it can process the SELECT statement. In this example, interactive SQL loads its metadata cache to allow it to check that the column EMPLOYEE_ID really exists for the table. The queries on the Oracle Rdb system tables RDB$RELATIONS and RDB$RELATION_FIELDS exceed the limit of rows.
  Oracle Rdb does not prepare the SELECT statement, let alone execute it. Raising the limit to a number less than 100 (the cardinality of EMPLOYEES) but more than the number of columns in EMPLOYEES (that is, the number of rows to read from the RDB$RELATION_FIELDS system table) is sufficient to read each column definition.
  To see an indication of the queries executed against the system tables, define the RDMS$DEBUG_FLAGS logical name or the RDB_DEBUG_FLAGS configuration parameter as S or B.
- If you set the row limit using the SQL SET QUERY statement and run the same query, it returns the number of rows specified by the SQL SET QUERY statement before failing:

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

  The SET QUERY LIMIT specifies that only user queries be limited to 10 rows. Therefore, the queries used to load the metadata cache are not restricted in any way.
  Like the SET QUERY LIMIT statement, the SQL precompiler and module processor command line qualifiers (QUERY_MAX_ROWS and SQLOPTIONS=QUERY_MAX_ROWS) only limit user queries.

Keep the differences in mind when limiting returned rows using the logical name RDMS$BIND_QG_REC_LIMIT or the configuration parameter RDB_BIND_QG_REC_LIMIT. They may limit more queries than are obvious. This is important when using 4GL tools, the SQL precompiler, the SQL

module processor, and other interfaces that read the Oracle Rdb system tables as part of query processing.

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

The CREATE INDEX process involves the following steps:

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

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

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

- You should commit the transaction immediately after the CREATE INDEX statement so that locks on the table are released. This avoids lock conflicts with other index definers and improves overall concurrency.
- By assigning the location of the temporary sort work files SORTWORK0, SORTWORK1, ..., SORTWORK9 to different disks for each parallel process that issues the SHARED DATA DEFINITION statement, you can increase the efficiency of sort operations. This minimizes any possible disk I/O bottlenecks and allows overlap of the SORT read/write cycle.
- If possible, enable global buffers and specify a buffer number large enough to hold a sufficient amount of table data. However, do not define global buffers larger than the available system physical memory. Global buffers allow sharing of database pages and thus result in disk I/O savings. That is, pages are read from disk by one of the processes and then shared by the other index definers for the same table, reducing the I/O load on the table.
- If global buffers are not used, ensure that enough local buffers exist to keep much of the index cached (use the RDM$BIND_BUFFERS logical name or RDB_BIND_BUFFERS configuration parameter or the NUMBER OF BUFFERS IS clause in SQL to change the number of buffers).
- To distribute the disk I/O load, place the storage areas for the indexes on separate disk drives. Note that using the same storage area for multiple indexes will result in contention during the index creation (Step 5) for SPAM pages.
- Consider placing the .ruj file for each parallel definer on its own disk or an infrequently used disk.
- Even though snapshot I/O should be minimal, consider disabling snapshots during parallel index creation.
- Refer to the *Oracle Rdb Guide to Performance and Tuning* to determine the appropriate working set values for each process to minimize excessive paging activity. In particular, avoid using working set parameters where the difference between WSQUOTA and WSEXTENT is large. The SORT utility uses the difference between these two values to allocate scratch virtual memory. A large difference (that is, the requested virtual memory grossly exceeds the available physical memory) may lead to excessive page faulting.
- The performance benefits of using SHARED DATA DEFINITION can best be observed when creating many indexes in parallel. The benefit is in the average elapsed time, not in CPU or I/O usage.

For example, when two indexes are created in parallel using the SHARED DATA DEFINITION clause, the database must be attached twice, and the two attaches each use separate system resources.
- Using the SHARED DATA DEFINITION clause on a single−file database or for indexes defined in the RDB$SYSTEM storage area is not recommended.

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

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

9.0.60 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 Rdb.

9.0.61 Nested Correlated Subquery Outer References Incorrect

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

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

The following example demonstrates the problem:

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

SQL> BEGIN
cont> FOR :A AS EACH ROW OF
cont>  SELECT * FROM VOYAGE V WHERE V.SHIP_NAME = 'SANDRA C.' OR
cont>                               V.SHIP_NAME = 'DAFFODIL' DO
cont>   FOR :B AS EACH ROW OF TABLE CURSOR MODCUR1 FOR
```

```
cont>    SELECT * FROM  MANIFEST M WHERE M.VOYAGE_NUM = :A.VOYAGE_NUM DO
cont>     UPDATE MANIFEST
cont>      SET TONNAGE = (SELECT (AVG (M1.EXP_NUM) *3) FROM MANIFEST M1
cont>                      WHERE M1.VOYAGE_NUM = :A.VOYAGE_NUM)
cont>      WHERE CURRENT OF MODCUR1;
cont>   END FOR;
cont> END FOR;
cont> END;
SQL> SELECT * FROM MANIFEST WHERE VOYAGE_NUM = 4904 OR
cont>                              VOYAGE_NUM = 4909;
  VOYAGE_NUM       EXP_NUM   MATERIAL              TONNAGE
        4904           311   CEDAR                    NULL
        4904           311   FIR                      NULL
        4909           291   IRON ORE                  933
        4909           350   BAUXITE                   933
        4909           350   COPPER                    933
        4909           355   MANGANESE                 933
        4909           355   TIN                       933
7 rows selected
```

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

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

For example:

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

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

9.0.62 Considerations when Using Holdable Cursors

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

before Oracle Rdb actually fetches them.

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

- If the access strategy forces Oracle Rdb to take a data snapshot, the data read and cached may be inaccurate by the time the cursor fetches the data.
  For example, user 1 opens a cursor and commits the transaction. User 2 deletes rows read by user 1 (this is possible because the read locks are released). It is possible for user 1 to report data now deleted and committed.
- If the access strategy uses indexes that allow duplicates, updates to the duplicates chain may cause rows to be skipped, or even revisited.
  Oracle Rdb keeps track of the dbkey in the duplicate chain pointing to the data that was fetched. However, the duplicates chain could be revised by the time Oracle Rdb returns to using it.

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

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

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

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

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

9.0.64 SQL Pascal Precompiler Processes ARRAY OF RECORD Declarations Incorrectly

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

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

```
main.spa:

    program main (input,output);

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

```
        begin
        end.

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

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

    good_def.pin

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

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

9.0.65 RMU Parallel Backup Command Not Supported for Use with SLS

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

9.1 Oracle CDD/Repository Restrictions

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

9.1.1 Oracle CDD/Repository Compatibility with Oracle Rdb Features

Some Oracle Rdb features are not fully supported by all versions of Oracle CDD/Repository. Table 9−1 shows which versions of Oracle CDD/Repository support Oracle Rdb features and the extent of support.

In Table 9−1, repository support for Oracle Rdb 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 Rdb operation to complete without aborting or overwriting metadata. With pass−through support, a CDD−I−MBLRSYNINFO informational message may be returned.

Table 9−1 Oracle CDD/Repository Compatibility for Oracle Rdb Features

Oracle Rdb Feature	Minimum Release of Oracle Rdb	Minimum Release of Oracle CDD/Repository	Support
CASE, NULLIF, and COALESCE expressions	6.0	6.1	Implicit
CAST function	4.1	7.0	Explicit
Character data types to support character sets	4.2	6.1	Implicit
Collating sequences	3.1	6.1	Explicit
Constraints (PRIMARY KEY, UNIQUE, NOT NULL, CHECK, FOREIGN KEY)	3.1	5.2	Explicit
CURRENT_DATE, CURRENT_TIME, and CURRENT_TIMESTAMP functions	4.1	7.0	Explicit
CURRENT_USER, SESSION_USER, SYSTEM_USER functions	6.0	7.0	Explicit
Date arithmetic	4.1	6.1	Pass−through
DATE ANSI, TIME, TIMESTAMP, and INTERVAL data types	4.1	6.1	Explicit
Delimited identifiers	4.2	6.1[1]	Explicit
External functions	6.0	6.1	Pass−through
External procedures	7.0	6.1	Pass−through
EXTRACT, CHAR_LENGTH, and OCTET_LENGTH functions	4.1	6.1	Explicit
GRANT/REVOKE privileges	4.0	5.0 accepts but does not store information	Pass−through
Indexes	1.0	5.2	Explicit
INTEGRATE DOMAIN	6.1	6.1	Explicit

INTEGRATE TABLE	6.1	6.1	Explicit
Logical area thresholds for storage maps and indexes	4.1	5.2	Pass−through
Multinational character set	3.1	4.0	Explicit
Multiversion environment (multiple Rdb versions)	4.1	5.1	Explicit
NULL keyword	2.2	7.0	Explicit
Oracle7 compatibility functions, such as CONCAT, CONVERT, DECODE, and SYSDATE	7.0	7.0	Explicit
Outer joins, derived tables	6.0	7.0	Pass−through
Query outlines	6.0	6.1	Pass−through
Storage map definitions correctly restored	3.0	5.1	Explicit
Stored functions	7.0	6.1	Pass−through
Stored procedures	6.0	6.1	Pass−through
SUBSTRING function	4.0	7.0 supports all features 5.0 supports all but 4.2 MIA features [2]	Explicit
Temporary tables	7.0	6.1	Pass−through
Triggers	3.1	5.2	Pass−through
TRUNCATE TABLE	7.0	6.1	Pass−through
TRIM and POSITION functions	6.1	7.0	Explicit
UPPER, LOWER, TRANSLATE functions	4.2	7.0	Explicit
USER function	2.2	7.0	Explict

[1]The repository does not preserve the distinction between uppercase and lowercase identifiers. If you use delimited identifiers with Oracle Rdb, the repository ensures that the record definition does not include objects with names that are duplicates except for case.
[2]Multivendor Integration Architecture (MIA) features include the CHAR_LENGTH clause and the TRANSLATE function.

9.1.2 Multischema Databases and CDD/Repository

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

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

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

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

Note

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

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

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

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

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

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

- While executing a CDO DEFINE REPOSITORY or DEFINE DICTIONARY command:
  - ♦ If the CDD$TEMPLATEDB backup (.rbf) file was created by a previous version of Oracle Rdb, the automatic RMU Convert operation that will be carried out on the .rbf file will fail because SYSPRV privilege is required.
  - ♦ If the CDD$TEMPLATEDB backup (.rbf) file was created by the current version of Oracle Rdb, 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 Rdb 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 Rdb is provided on the Oracle Rdb kit for use on OpenVMS VAX systems. See Section 9.1.3.1 for details.

9.1.3.1 Installing the Corrected CDDSHR Images

OpenVMS VAX Systems

Note

The following procedure must be carried out if you have installed or plan to install Oracle Rdb 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 Rdb for OpenVMS VAX distribution kit is a CDD upgraded image kit, called CDDRDB042, that must be installed after you have installed the Oracle Rdb 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 Rdb 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.

9.1.3.2 CDD Conversion Procedure

OpenVMS VAX Systems

Oracle Rdb 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
```

| Contents