Oracle® Rdb for OpenVMS

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Oracle® Rdb for OpenVMS

Release Notes

Release 7.0.8

April 2005

Oracle Rdb Release Notes, Release 7.0.8 for OpenVMS

Contents

Preface

Purpose of This Manual

This manual contains release notes for Oracle Rdb Release 7.0.8. 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 7.0.8.

Document Structure

This manual consists of nine chapters:

Chapter 1	Describes how to install Oracle Rdb Release 7.0.8.
Chapter 2	Describes software errors corrected in Oracle Rdb Release 7.0.8.
Chapter 3	Describes software errors corrected in Oracle Rdb Release 7.0.7.3.
Chapter 4	Describes software errors corrected in Oracle Rdb Release 7.0.7.2.
Chapter 5	Describes software errors corrected in Oracle Rdb Release 7.0.7.1.
Chapter 6	Describes enhancements introduced in Oracle Rdb Release 7.0.8.
Chapter 7	Describes enhancements introduced in previous releases of Oracle Rdb.
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 7.0.8.

Chapter 1
Installing Oracle Rdb Release 7.0.8

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 130,000 free blocks on your system disk for OpenVMS VAX systems; 240,000 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 7.0.8 are:

- RDBSH070 for Oracle Rdb for OpenVMS VAX standard version.
- RDBASH070 for Oracle Rdb for OpenVMS Alpha standard version.
- RDBMVH070 for Oracle Rdb for OpenVMS VAX multiversion.
- RDBAMVH070 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:[RDBAMVH070.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 multiversion kit on device MTA0: and print the release notes:

```
$ @SYS$UPDATE:VMSINSTAL RDBAMVH070 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, "RDBVMSV7.0−8".

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=RDBVMSV7.0−8 −
_$ /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 8.2−x 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. As of Oracle Rdb Release 7.0.7, the Alpha EV7 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 7.0.8

This chapter describes software errors that are fixed by Oracle Rdb Release 7.0.8.

2.1 Software Errors Fixed That Apply to All Interfaces

2.1.1 Bugcheck Dump During ALTER STORAGE MAP Command

Bug 2825363

Under extremely rare circumstances, it was possible that an ALTER STORAGE MAP command could cause various bugchecks due to memory corruption. For this to happen, the ALTER STORAGE MAP command must be causing rows to be moved, and an index must exist that has duplicate values and is of TYPE IS SORTED RANKED.

In the following example, the storage map is altered to move the rows in the table to a new set of storage areas. A sorted ranked index exists with many duplicate values so this index needs to be updated to reflect the new location of each record.

```
SQL> alter storage map DATA_MAP
cont>  partitioning is not updatable
cont>  store using (MY_ID)
cont>  in DATA_AREA_1  (threshold is (83)) with limit of (1)
cont>  in DATA_AREA_2  (threshold is (83)) with limit of (2)
cont>  in DATA_AREA_3  (threshold is (83)) with limit of (3)
cont>  otherwise in  DATA_AREA;
%RDMS-I-BUGCHKDMP, generating bugcheck dump file DISK1:[TEST]RDSBUGCHK.DMP;
%RDMS-I-BUGCHKDMP, generating bugcheck dump file DISK1:[TEST]SQLBUGCHK.DMP;
```

The exception in the bugcheck is an access violation and usually it is in one of the memory management routines such as COSI_MEM_GET_VM or COSI_MEM_FREE_VM.

There are three ways to avoid the problem:

- The offending index can be dropped prior to altering the storage map and recreated afterwards.
- The table can be unloaded and truncated prior to altering the storage map. The data can then be sucessfully loaded when the alter process completes.
- If the index is rebuilt with a fullness threshold less than 100%, the problem should not occur.

The problem is extremely rare and does not cause corruption to the data or index. If the problem occurs, the database is automatically recovered, and one of the methods described above can be used to avoid the problem.

This problem has been corrected in Oracle Rdb Release 7.0.8.

2.1.2 Unexpected Constraint Failures from RMU/Verify/Constraint

Bugs 2303741 and 2649462

In prior releases of Oracle Rdb, it was possible in rare cases to see one of the following commands report a failure while validating referential integrity constraints. Repeating the action was often successful.

- RMU Verify/Constraint
- ALTER TABLE ... ENABLE CONSTRAINTS
- TRUNCATE TABLE

This problem was caused by a timing condition in the table verify code. The following example shows one of the errors:

```
SQL> Alter table degrees enable all constraints;
%RDMS−I−BUGCHKDMP, generating bugcheck dump file DISK1:[TEST]RDSBUGCHK.DMP;
```

The bugcheck summary looks similar to this:

- Alpha OpenVMS 7.3−2
- Oracle Rdb Server X7.1−00
- Got a RDSBUGCHK.DMP
- SYSTEM−F−ILLEGAL_SHADOW, illegal formed trap shadow, Imask=05AC5DCC, Fmask=0000001B, summary=C0, PC=0000000000000000, PS=00000000
- Exception occurred at symbol not found
- Database root: DISK1:[TEST.CONSTRAINTS]MF_PERSONNEL_SQL

The RMU Verify/Constraints procedure may simply report a constraint failure (RMU−I−CONSTFAIL, Verification of constraint "<name>" has failed). Defining the logical name RDMS$SET_FLAGS and assigning the value ITEM_LIST will cause RMU to report the failure status of the constraint verification. This may include the following:

```
~H: ...verify constraint "SOMETABLE_SOMECOLUMN_NOT_NULL4"
~H: ...verify complete with failure status 000005B4

Status 000005B4 is "%SYSTEM−F−ILLEGAL_SHADOW, illegal formed trap shadow,
Imask=<hex>, Fmask=<hex>,  summary=!XB,
PC=<hex>, PS=<hex>".

~H: ...verify constraint "THISTABLE_THATCOLUMN_NOT_NULL10"
~H: ...verify complete with failure status 0000000C

Status 0000000C is "SYSTEM−F−ACCVIO, access violation, reason mask=!XB,
virtual address=<hex>,  PC=<hex>, PS=<hex>".
```

This problem has been corrected in Oracle Rdb Release 7.0.8.

2.1.3 Redundant Index in Dynamic Index Only Tactic

Bug 3439578

Where multiple indexes appear useful for retrieving data for a query, and at least one of the indexes contains all columns referenced by the query, the dynamic optimizer may scan additional indexes even though the conditions used had been used on a previous index lookup. This resulted in wasted I/O and poor performance.

In the following example, the columns used for both index lookups on table TTOS270 are identical, but in a different order.

```
~S#0003
Tables:
```

```
  0 = TTOS275
  1 = TTOS270
Firstn: 100000
Cross block of 2 entries
  Cross block entry 1
    Conjunct: (0.VALIDITY_DATE_FROM < DATE '2001-08-09') AND (SUBSTRING (
              0.GROUP_CODE FROM (1 - 1) FOR 2) = 'SG')
    Index only retrieval of relation 0:TTOS275
      Index name  P_TTOS275 [0:0]
  Cross block entry 2
    Conjunct: <agg0> = 0
   Aggregate-F1: 0:COUNT-ANY (<subselect>)
    Leaf#01 NdxOnly 1:TTOS270 Card=1290000
      Bool: (0.ACCOMM_CODE = 1.ACCOMM_CODE) AND (0.UNIT_TYPE_CODE =
            1.UNIT_TYPE_CODE) AND (0.GROUP_CODE = 1.GROUP_CODE) AND (
            0.ALLOCATION_PRODUCT = 1.ALLOCATION_PRODUCT) AND (
            0.VALIDITY_DATE_FROM >= 1.VALIDITY_DATE_FROM) AND (
            0.VALIDITY_DATE_FROM <= 1.VALIDITY_DATE_TO)
      FgrNdx  P_TTOS270 [4:5] Fan=15
        Keys: (0.GROUP_CODE = 1.GROUP_CODE) AND (0.ALLOCATION_PRODUCT =
              1.ALLOCATION_PRODUCT) AND (0.ACCOMM_CODE = 1.ACCOMM_CODE) AND (
              0.UNIT_TYPE_CODE = 1.UNIT_TYPE_CODE) AND (0.VALIDITY_DATE_FROM >=
              1.VALIDITY_DATE_FROM)
      BgrNdx1 S_TTOS270 [4:5] Fan=15
        Keys: (0.ACCOMM_CODE = 1.ACCOMM_CODE) AND (0.UNIT_TYPE_CODE =
              1.UNIT_TYPE_CODE) AND (0.GROUP_CODE = 1.GROUP_CODE) AND (
              0.ALLOCATION_PRODUCT = 1.ALLOCATION_PRODUCT) AND (
              0.VALIDITY_DATE_FROM >= 1.VALIDITY_DATE_FROM)
```

Oracle Rdb now detects this and does not use the redundant index. This results in a static index-only tactic with one index, as shown in the following example.

```
~S#0003
Tables:
  0 = TTOS275
  1 = TTOS270
Firstn: 100000
Cross block of 2 entries
  Cross block entry 1
    Conjunct: (0.VALIDITY_DATE_FROM < DATE '2001-08-09') AND (SUBSTRING (
              0.GROUP_CODE FROM (1 - 1) FOR 2) = 'SG')
    Index only retrieval of relation 0:TTOS275
      Index name  P_TTOS275 [0:0]
  Cross block entry 2
    Conjunct: <agg0> = 0
    Aggregate-F1: 0:COUNT-ANY (<subselect>)
    Conjunct: 0.VALIDITY_DATE_FROM <= 1.VALIDITY_DATE_TO
    Index only retrieval of relation 1:TTOS270
      Index name  P_TTOS270 [4:5]
        Keys: (0.GROUP_CODE = 1.GROUP_CODE) AND (0.ALLOCATION_PRODUCT =
              1.ALLOCATION_PRODUCT) AND (0.ACCOMM_CODE = 1.ACCOMM_CODE) AND (
              0.UNIT_TYPE_CODE = 1.UNIT_TYPE_CODE) AND (0.VALIDITY_DATE_FROM >=
              1.VALIDITY_DATE_FROM)
```

The problem can be avoided by using the RDMS$MAX_STABILITY logical name to disable the dynamic optimizer.

This problem has been corrected in Oracle Rdb Release 7.0.8.

2.1.4 Buffer Overflow in Code Generated by RDMS$$STITM_FROM_CACT

Bugs 3354321, 3566244 and 3589230

On rare occasions, an ACCVIO and a bugcheck would occur because of a buffer overflow condition. Code generated by RDMS$$STITM_FROM_CACT may have tried to read beyond its data buffer.

This problem has been corrected in Oracle Rdb Release 7.0.8.

2.1.5 Intermittent Bugchecks on KOD$ROLLBACK or KOD$PREPARE

Bug 3902286

When you used SQL*Plus, intermittant bugchecks could occur if lock conflicts were present. This was due to an error in hold cursor processing.

The following shows examples of bugchecks:

```
COSI-F-BUGCHECK, internal consistency failure
Exception occurred  at KOD$ROLLBACK + 00000258
Called from RDMS$$INT_ROLLBACK_TRANSACTION + 0000055C
Called from RDMS$TOP_ROLLBACK_TRANSACTION + 000003EC
Called from BLI$CALLG + 000000BC

COSI-F-BUGCHECK, internal consistency failure
Exception occurred at KOD$PREPARE + 00000228
Called from KOD$COMMIT + 0000018C
Called from RDMS$$INT_COMMIT_TRANSACTION + 000002BC
Called from RDMS$TOP_COMMIT_TRANSACTION + 0000021C
```

This problem has been corrected in Oracle Rdb Release 7.0.8.

2.1.6 NOREQIDT Error After Many Attaches or Detaches

Bug 3971379

An application that utilized lock timeouts (for example, the WAIT clause of the SET TRANSACTION statement) and often disconnected from a database could eventually encounter the following error:

```
%RDMS-F-NOREQIDT, reached internal maximum number of simultaneous timer requests
```

This problem could be reproduced by creating two login sessions. The first session would lock a row in a table. The second session would repeatedly attempt to access the same row. After each attempt it would disconnect from the database and attach again. For example:

Session 1:

```
SQL> ATTACH 'FILE MF_PERSONNEL';
SQL> UPDATE EMPLOYEES SET SEX = SEX WHERE EMPLOYEE_ID = '00164';
```

Session 2:

Repeat the following sequence of commands without exiting SQL:

```
SQL> ATTACH 'FILENAME MF_PERSONNEL';
SQL> SET TRANSACTION READ WRITE WAIT 1;
SQL> UPDATE EMPLOYEES SET SEX = SEX WHERE EMPLOYEE_ID = '00164';
SQL> ROLLBACK;
SQL> DISCONNECT ALL;
```

After about 100 iterations, the UPDATE command would fail with the NOREQIDT error.

You can avoid this problem by not repeatedly detaching from and attaching to the database within a single invocation of a database application, or by not using the lock timeout feature.

This problem has been corrected in Oracle Rdb Release 7.0.8.

2.1.7 Processes Hang in HIB State

Bugs 2881846 and 3807295

Oracle Rdb applications that hibernate may occasionally hang in HIB state if after−image journaling (AIJ) is enabled. Note that applications utilizing the OpenVMS POSIX Threads Library will often hibernate, so threaded applications are especially vulnerable to this issue.

This problem could occur if a user application had executed the OpenVMS $HIBER system service and one of the following Oracle Rdb events occurred while the process was hibernating:

- A Global Checkpoint request was issued by a journal switch or an RMU Checkpoint command. (This issue was resolved in Releases 7.0.7.2 and 7.1.2.2; see Section 4.1.2 for more information.)
- A checkpoint timer expired.
- A two−phase commit (2PC or DECdtm) transaction that involved the process was ended.

Note that if the OpenVMS POSIX Threads Library is being utilized, the OpenVMS remedial kit VMS732_SYS−V0600 may also be required to completely resolve this issue. Have your OpenVMS support specialist reference CFS numbers 108102, 108308 and 108322.

This problem can be avoided by disabling journaling. Oracle Corporation does not recommend disabling journaling unless another suitable database recovery method is available.

This problem has been corrected in Oracle Rdb Release 7.0.8.

2.1.8 Database Corruption When 2PC Transaction Fails

Bug 4011078

It was possible for a database to become corrupt when the following conditions were true:

- The FAST COMMIT feature was enabled.
- A failing process was a participant in a two−phase commit (2PC) transaction.

- The process failed after completing the prepare phase of a 2PC transaction but before completing the commit phase.
- Other participants in the same transaction failed before completing the prepare phase.

In this situation, the database recovery process (DBR) would neglect to rollback the failed transaction.

This problem was introduced in Releases 7.0.7.1 and 7.1.2.

There is no workaround for this problem.

This problem has been corrected in Oracle Rdb Release 7.0.8.

2.1.9 Bugchecks at DIOMARK$NEW_SNAP_PAGE + 000000D0 When Area Added Online

Bug 3818408

If a database was currently open on multiple nodes and a storage area was added on one node, attempts to reference that area on other nodes could result in a bugcheck containing an exception similar to the following:

```
***** Exception at 011F4500 : DIOMARK$NEW_SNAP_PAGE + 000000D0
%SYSTEM-F-ACCVIO, access violation, reason mask=00,
  virtual address=0000000000000008, PC=00000000011F4500, PS=0000000B
```

This problem can be demonstrated with the following commands:

Session on node 1:

```
$ SQL$
SQL> CREATE DATABASE FILENAME TEST OPEN IS MANUAL
SQL>   RESERVE 10 STORAGE AREAS
SQL>
SQL>   CREATE STORAGE AREA RDB$SYSTEM FILENAME TEST;
SQL> EXIT
```

Session on node 2:

```
$ RMU/OPEN/WAIT TEST
```

Session on node 1:

```
$ RMU/OPEN/WAIT TEST
$
$ SQL$
SQL> ALTER DATABASE FILENAME TEST
SQL>   ADD STORAGE AREA AREA1 FILENAME AREA1;
SQL>
SQL> ATTACH 'FILENAME TEST';
SQL> CREATE TABLE T1 (F1 INTEGER);
SQL> CREATE STORAGE MAP M1 FOR T1 STORE IN AREA1;
SQL> COMMIT;
SQL> EXIT
```

Session on node 2:

```
SQL> ATTACH 'FILENAME TEST';
SQL> INSERT INTO T1 VALUES (1);
%RDMS-I-BUGCHKDMP, generating bugcheck dump file dev:[dir]RDSBUGCHK.DMP;
%RDMS-I-BUGCHKDMP, generating bugcheck dump file dev:[dir]SQLBUGCHK.DMP;
%SYSTEM-F-ACCVIO, access violation, reason mask=00,
virtual address=0000000000000008, PC=000000000027F20C, PS=0000001B
SQL> ROLLBACK;
SQL> EXIT;
```

The bugcheck would only occur if a specific sequence of actions occurred the first time the new storage area was accessed on a node that did not create the area. Under certain conditions, the data structure describing the new storage area was not being read from disk prior to being accessed.

The problem can be avoided by closing the database on all nodes prior to adding a new storage area.

This problem has been corrected in Oracle Rdb Release 7.0.8.

2.1.10 Select Count Query With Host Variable Returns Wrong Result

Bug 4208119

The following select count query, which contains a WHERE clause with a host variable, returns the wrong result:

```
declare :x integer;
begin
set :x = 1;
set :x = 1;
end;

sel count(*) from employees where :x = 2;
Tables:
  0 = EMPLOYEES
Aggregate: 0:COUNT (*)
Index only retrieval of relation 0:EMPLOYEES
  Index name  EMP_EMPLOYEE_ID [0:0]

         100
1 row selected
```

The query works if the host variable is removed.

```
sel count(*) from employees where 1 = 2;
Tables:
  0 = EMPLOYEES
Aggregate: 0:COUNT (*)
Conjunct: 1 = 2
Index only retrieval of relation 0:EMPLOYEES
  Index name  EMP_EMPLOYEE_ID [0:0]

           0
1 row selected
```

The query also works if the count is removed.

```
sel * from employees where :x = 2;
Tables:
  0 = EMPLOYEES
Leaf#01 FFirst 0:EMPLOYEES Card=10000
  Bool: <var0> = 2
  BgrNdx1 EMP_EMPLOYEE_ID [0:0] Fan=17
    Bool: <var0> = 2
0 rows selected
```

This problem has been corrected in Oracle Rdb Release 7.0.8.

2.1.11 UNION Join Query With Host Variable in the Predicate Returns Wrong Result

Bug 4216829

The following UNION join query with host variable in the predicate returns the wrong result:

```
declare  :i_batch_id integer;
declare  :i_customer varchar(5);
begin
set :i_batch_id = 1;
set :i_customer = 'USCC';
END;

select m.batch_id, m.series_country
from ent_marg_dtl as m
     inner join (
        select  series_country,
                series_market
        from ent_series
        where series_market in (2, 20)
        union
        select  series_country,
                series_market
        from ent_series
        where series_market not in (2, 20)
        ) as s
        on      m.series_country = s.series_country
        and     m.series_market = s.series_market
where   m.batch_id = :i_batch_id
        and (m.customer = :i_customer
             or :i_customer = '') ;             <== see Note
Tables:
  0 = ENT_MARG_DTL
  1 = ENT_SERIES
  2 = ENT_SERIES
Cross block of 2 entries
  Cross block entry 1
    Leaf#01 FFirst 0:ENT_MARG_DTL Card=1
      Bool: (0.BATCH_ID = <var0>) AND ((0.CUSTOMER =
<var1>) OR (<var2> = ''))
      BgrNdx1 ENT_MARG_DTL_IDX01 [1:1] Fan=15
        Keys: 0.BATCH_ID = <var0>
  Cross block entry 2
    Conjunct: (0.SERIES_COUNTRY = <mapped field>) AND
```

```
 (0.SERIES_MARKET =  <mapped field>)
    Merge of 1 entries
      Merge block entry 1
      Reduce: <mapped field>, <mapped field>
      Sort: <mapped field>(a), <mapped field>(a)
      Merge of 2 entries
        Merge block entry 1
        Conjunct: (1.SERIES_MARKET = 2) OR (1.SERIES_MARKET = 20)
        Index only retrieval of relation 1:ENT_SERIES
          Index name  ENT_SERIES_IDX01 [0:0]
        Merge block entry 2
        Conjunct: <var2> = ''                    <== see Note
        Conjunct: 0.BATCH_ID = <var0>
        Conjunct: (2.SERIES_MARKET <> 2) AND (2.SERIES_MARKET <> 20)
        Index only retrieval of relation 2:ENT_SERIES
          Index name  ENT_SERIES_IDX01 [0:0]
0 rows selected
```

Note: The conjunct ":i_customer = "" is pushed down to the second leg of the UNION clause. Since this is the right side operand of the OR predicate it cannot be stripped out and pushed down without the other operand.

The query works if the host variable in one of the predicates is replaced by the text literal 'USCC'.

```
select m.batch_id, m.series_country
from ent_marg_dtl as m
     inner join (
        select  series_country,
                series_market
        from ent_series
        where series_market in (2, 20)
        union
        select  series_country,
                series_market
        from ent_series
        where series_market not in (2, 20)
        ) as s
        on      m.series_country = s.series_country
        and     m.series_market = s.series_market
    where   m.batch_id = :i_batch_id
        and (m.customer = :i_customer
             or 'USCC' = ''                      ! <== replaced with 'USCC'
             ) ;
Tables:
  0 = ENT_MARG_DTL
  1 = ENT_SERIES
  2 = ENT_SERIES
Cross block of 2 entries
  Cross block entry 1
    Leaf#01 FFirst 0:ENT_MARG_DTL Card=1
      Bool: (0.BATCH_ID = <var0>) AND ((0.CUSTOMER =
 <var1>) OR ('USCC' = ''))
      BgrNdx1 ENT_MARG_DTL_IDX01 [1:1] Fan=15
        Keys: 0.BATCH_ID = <var0>
  Cross block entry 2
    Conjunct: (0.SERIES_COUNTRY = <mapped field>) AND
 (0.SERIES_MARKET = <mapped field>)
    Merge of 1 entries
      Merge block entry 1
      Reduce: <mapped field>, <mapped field>
      Sort: <mapped field>(a), <mapped field>(a)
```

```
      Merge of 2 entries
        Merge block entry 1
        Conjunct: (1.SERIES_MARKET = 2) OR (1.SERIES_MARKET = 20)
        Index only retrieval of relation 1:ENT_SERIES
          Index name  ENT_SERIES_IDX01 [0:0]
        Merge block entry 2
        Conjunct: 0.BATCH_ID = <var0>           <== see Note1
        Conjunct: (2.SERIES_MARKET <> 2) AND (2.SERIES_MARKET <> 20)
        Index only retrieval of relation 2:ENT_SERIES
          Index name  ENT_SERIES_IDX01 [0:0]
  M.BATCH_ID   M.SERIES_COUNTRY
           1                12
1 row selected
```

Note1: Note that the conjunct "'USCC' = "" does not appear in the second leg of the UNION clause.

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

1. The main query selects from inner−joining a table with a UNION of two sub−select queries.
2. The WHERE clause contains an OR predicate referencing a host variable twice in the equality filters.

There is no known workaround for this problem.

This problem has been corrected in Oracle Rdb Release 7.0.8.

2.1.12 Bugcheck in COSI_MEM_FREE_VMLIST After Update of Ranked Index

Bug 4191015

During updates on indexes of *TYPE IS SORTED RANKED*. it was possible that memory could be overwritten and memory queues become corrupt. When the memory queue was subsequently processed, a bugcheck would result.

In the following example, the update statement left a memory queue in this corrupt state. During exit from SQL, while detaching from the database, the memory queue is processed to free up the memory and a bugcheck results.

```
SQL> update some_table set some_key=626 where another_key=624;
32419 rows updated
SQL> commit;
SQL> exit
%RDMS−I−BUGCHKDMP, generating bugcheck dump file MBRADLEY_USR:[BRADLEY]
RDSBUGCHK.DMP
$
```

The following example shows the exception and the first few calls from the bugcheck.

```
%SYSTEM−F−ACCVIO, access violation, reason mask=04, virtual
address=0000000050000004, PC=00000000011D68E4, PS=0000000B

Saved PC = 0115C214 : RDMS$$RDMSCHEMA_UNLOAD_META + 00000434
Saved PC = 0115CB5C : RDMS$$RDMSCHEMA_UNLOAD_TABLE + 0000024C
Saved PC = 0117FCD4 : RDMS$$DETACH_DATABASE + 000002F4
```

```
Saved PC = 0117F944 : RDMS$TOP_DETACH_DATABASE + 00000424
```

In the reported case, the update completed sucessfully and the index was not corrupt in any way.

There is no workaround for this problem.

This problem has been corrected in Oracle Rdb Release 7.0.8.

2.1.13 Spurious SYSVERDIF Message During Installation

When installing Oracle Rdb on an OpenVMS V8.2 Alpha system, depending on the previous version of Oracle Rdb installed and how Oracle Rdb was shut down prior to the installation, a message similar to the following may be displayed during the installation procedure:

```
%INSTALL−E−FAIL, failed to REPLACE entry for
DISK$VMS82:<SYS0.SYSCOMMON.SYSLIB>RDMXSMP70.EXE
−INSTALL−E−SYSVERDIF, system version mismatch − please relink
```

This message may be safely ignored. Using the RMONSTOP.COM (for standard installations) or RMONSTOP70.COM (for multi−version installations) procedure to shut down Oracle Rdb prior to the installation may help avoid the message.

2.1.14 Query with a Shared Predicate in OR Statement Returns Wrong Result

Bug 3918278

The following query with a shared predicate in an OR statment should return two rows, but it returns no rows.

```
SELECT
   t1.security_id,
   t1.quotation_basis_code
 FROM
   security_quotation t1,
   quotation_basis t2,
   (SELECT c1.security_id, c1.issuer_id FROM equity c1
    union
    SELECT c2.security_id, c2.issuer_id FROM fixed_interest c2) AS t3,
   (SELECT
        c3.security_id,
        c3.chess_eligibility_code AS chess_eligibility_code,
        c3.first_listed_date AS first_listed_date,
        c3.last_listed_date AS last_listed_date
      from
          equity c3 LEFT OUTER JOIN equity_tran c4
          on c3.security_id = c4.security_id
   ) AS t4
 WHERE
    t1.quotation_basis_start_date = '09−SEP−2004'  AND
    t1.quotation_basis_code = t2.quotation_basis_code AND
    t1.security_id          = t3.security_id AND
    t1.security_id          = t4.security_id AND
(
  ((t4.last_listed_date >= '08−SEP−2004' OR
    t4.last_listed_date = '17−NOV−1858') AND
```

```
      t1.quotation_basis_code <> 'RE'
      )
    OR                          ! <== OR statement with shared predicates
  ((t4.last_listed_date >= '09-SEP-2004' OR
    t4.last_listed_date = '17-NOV-1858'  ) AND
      t1.quotation_basis_code = 'RE'
      )
   OR
  (t4.chess_eligibility_code > 0)
  )
 ;
Tables:
  0 = SECURITY_QUOTATION
  1 = QUOTATION_BASIS
  2 = EQUITY
  3 = FIXED_INTEREST
  4 = EQUITY
  5 = EQUITY_TRAN
Cross block of 3 entries
  Cross block entry 1
    Conjunct: 0.SECURITY_ID = 4.SECURITY_ID
    Conjunct: (((4.LAST_LISTED_DATE >= '8-SEP-2004') OR (4.LAST_LISTED_DATE =
              '17-NOV-1858')) AND (0.QUOTATION_BASIS_CODE <> 'RE')) OR (((
              4.LAST_LISTED_DATE >= '9-SEP-2004') OR (4.LAST_LISTED_DATE =
              '17-NOV-1858')) AND (0.QUOTATION_BASIS_CODE = 'RE')) OR (
              4.CHESS_ELIGIBILITY_CODE > 0)
    Match
      Outer loop
        Merge of 1 entries
          Merge block entry 1
          Match    (Left Outer Join)
            Outer loop
              Get     Retrieval by index of relation 4:EQUITY
                Index name  EQY_U_PRM [0:0]
            Inner loop      (zig-zag)
              Index only retrieval of relation 5:EQUITY_TRAN
                Index name  EQT_U_PRM [0:0]
      Inner loop      (zig-zag)
        Conjunct: 0.QUOTATION_BASIS_START_DATE = '9-SEP-2004'
        Index only retrieval of relation 0:SECURITY_QUOTATION
          Index name  SQB_U_PRM [0:0]
  Cross block entry 2
    Index only retrieval of relation 1:QUOTATION_BASIS
      Index name  QTB_U_PRM [1:1]    Direct lookup
        Keys: 0.QUOTATION_BASIS_CODE = 1.QUOTATION_BASIS_CODE
  Cross block entry 3
    Conjunct: (0.SECURITY_ID = <mapped field>) AND (0.SECURITY_ID =
              4.SECURITY_ID) AND ((4.LAST_LISTED_DATE >= '8-SEP-2004') OR (
              4.LAST_LISTED_DATE >= '9-SEP-2004') OR (4.CHESS_ELIGIBILITY_CODE
               > 0))
    Merge of 1 entries
      Merge block entry 1
      Reduce: <mapped field>, <mapped field>
      Sort: <mapped field>(a), <mapped field>(a)
      Merge of 2 entries
        Merge block entry 1
        Get Retrieval by index of relation 2:EQUITY
          Index name  EQY_U_PRM [1:1] Direct lookup
            Keys: 0.SECURITY_ID = <mapped field>
        Merge block entry 2
        Get     Retrieval by index of relation 3:FIXED_INTEREST
```

```
          Index name  FIN_U_PRM [1:1]    Direct lookup
            Keys: 0.SECURITY_ID = <mapped field>
0 rows selected
```

The first appearance of the conjuncts created in the above query represents the complete OR statement.

```
    Conjunct: 0.SECURITY_ID = 4.SECURITY_ID
    Conjunct: (((4.LAST_LISTED_DATE >= '8-SEP-2004') OR (4.LAST_LISTED_DATE =
              '17-NOV-1858')) AND (0.QUOTATION_BASIS_CODE <> 'RE')) OR (((
              4.LAST_LISTED_DATE >= '9-SEP-2004') OR (4.LAST_LISTED_DATE =
              '17-NOV-1858')) AND (0.QUOTATION_BASIS_CODE = 'RE')) OR (
              4.CHESS_ELIGIBILITY_CODE > 0)
```

However, the next appearance is incomplete: the shared predicate "t4.last_listed_date = '17−NOV−1858'" is missing from both the left and right sides of the OR statement.

```
    Conjunct: (0.SECURITY_ID = <mapped field>) AND (0.SECURITY_ID =
              4.SECURITY_ID) AND ((4.LAST_LISTED_DATE >= '8-SEP-2004') OR (
              4.LAST_LISTED_DATE >= '9-SEP-2004') OR (4.CHESS_ELIGIBILITY_CODE
               > 0))
```

Also missing are the predicates "t1.quotation_basis_code <> 'RE'" and "t1.quotation_basis_code = 'RE'".

The only workaround is to swap the left and right side terms of the OR statement but this is not acceptable as a real workaround due to the changes to the SQL scripts.

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

- The query joins several tables, including derived tables from a UNION join and left OUTER join.
- The WHERE clause contains an OR statement with shared predicates in the left and right side terms.
- The OR statement also contains a filter predicate on both the left and right side terms.

This problem has been corrected in Oracle Rdb Release 7.0.8.

2.1.15 Query with Dbkey Retrieval Returns Wrong Result

Bug 3878628

The following query should retrieve the desired row by the specified dbkey.

```
declare :dbk bigint;
select dbkey into :dbk from T1 where TRAN_SEQ_NBR = 350;
select tran_seq_nbr, RC_ID
!       ,SEGMENT_NBR, PROC_TIMESTAMP
 from T1 where dbkey = :dbk;
Tables:
  0 = T1
Leaf#01 FFirst 0:T1 Card=11
  Bool: 0.DBKEY = <var0>
  BgrNdx1 T1_NDX [0:0] Fan=30
    Bool: 0.DBKEY = <var0>
0 rows selected
```

The query works if the old cost model is enabled by setting SQL flags 'OLD_COST_MODEL'.

```
set flags 'OLD_COST_MODEL';
select tran_seq_nbr, rc_id from T1 where dbkey = :dbk;
Tables:
  0 = T1
Conjunct: 0.DBKEY = <var0>
Firstn: 1
Get     Retrieval by DBK of relation 0:T1
 TRAN_SEQ_NBR   RC_ID
          350   0211
1 row selected
```

This problem only happens when the row is very large compared to the page size. Oracle Rdb assumed that the row was fragmented over several pages and thus avoided applying the dbkey retrieval strategy.

A workaround would be to move this table to an area with larger pages so that fragmentation does not occur. Also this would not happen on larger tables where the index depth is greater.

Using SET FLAGS 'MAX_STABILITY' can also be used to eliminate the estimate step in the dynamic optimizer but the resultant strategy is a full scan index retrieval which is not as efficient as the dbkey retrieval strategy.

```
set flags 'MAX_STABILITY';

select * from T1 where dbkey = :dbk;
Tables:
  0 = T1
Get     Retrieval by index of relation 0:T1
  Index name  T1_NDX [0:0]
    Bool: 0.DBKEY = <var0>
 ...etc...
1 row selected
```

This problem has been corrected in Oracle Rdb Release 7.0.8.

2.1.16 Complex Query with MAX()...GROUP BY Returns Wrong Result

Bug 3719771

A complex nested SQL query with MAX()...GROUP BY clause returned the wrong result (1 row selected) when the cardinality of the table was increased to 2.1 million rows. If the SQL flag 'MAX_STABILITY' was enabled, the query returned the correct result which is 65 rows out of 2094 rows.

The following example query shows the problem where the main select returns columns from the two main subselects, Q2 and Q5.

The problem occurs in the Q5 subselect clause. Note that the subselect Q5 itself is made up of two nested subselects, Q67 and Q89, which are nearly identical sub−queries except for the WHERE clauses. The query returns the incorrect result if the Q67 columns are selected for Q5, but it returns the correct result if the Q89 columns are selected instead.

```
create data file test_bug;

create table TAB1 (
```

```
C01_ID                    CHAR(11)       ,
C05_ID                    CHAR(1)        ,
C02_LOC                   CHAR(2)        ,
OPERATION_NO              CHAR(5)        ,
FINISH_DATIME             TIMESTAMP(2)   ,
C02_PRODUCT_ID            CHAR(3)        ,
C08_CODE                  CHAR(1)        ,
C06_COUNT                 INTEGER        ,
C07_COUNT                 INTEGER
);

INSERT INTO TAB1 (C01_ID, C05_ID, C02_LOC, OPERATION_NO, FINISH_DATIME,
    C02_PRODUCT_ID, C08_CODE, C06_COUNT, C07_COUNT) VALUE
  ('1K46804424', '2', '20', '51000', timestamp'2004-06-01 20:17:31.00',
   'DAB', '0', 1, 0);


INSERT INTO TAB1 (C01_ID, C05_ID, C02_LOC, OPERATION_NO, FINISH_DATIME,
    C02_PRODUCT_ID, C08_CODE, C06_COUNT, C07_COUNT) VALUE
  ('1K46804420', '2', '20', '51000', timestamp'2004-06-01 20:15:38.00',
   'DAB', '0', 1, 0);

INSERT INTO TAB1 (C01_ID, C05_ID, C02_LOC, OPERATION_NO, FINISH_DATIME,
    C02_PRODUCT_ID, C08_CODE, C06_COUNT, C07_COUNT) VALUE
  ('1K46804417', '2', '20', '51000', timestamp'2004-06-01 20:16:28.00',
   'DAB', '0', 1, 0);

create unique index TAB1_IDX_S1 on TAB1 (
    C01_ID, C02_LOC, OPERATION_NO, FINISH_DATIME);

create unique index TAB1_IDX_S3 on TAB1 (
    OPERATION_NO, C01_ID, C05_ID, C02_LOC, FINISH_DATIME);

create unique index TAB1_IDX_S4 on TAB1 (
    FINISH_DATIME, C01_ID, C05_ID, C02_LOC, OPERATION_NO);

commit;
disconnect all;

attach file 'test_bug';
set flags 'strategy';
select
    Q2.C01_ID,
    Q2.C02_LOC,
    Q5.C01_ID
  from
     (select
          Q3.C01_ID,
          Q3.C02_LOC,
          Q3.OPERATION_NO,
          T1.C07_COUNT
      from
         (select
              Q4.C01_ID,
              Q4.C02_LOC,
              Q4.OPERATION_NO,
              Q4.C07_COUNT
          from
             (select
                  C01_ID,
                  C02_LOC,
```

```
            OPERATION_NO,
            C07_COUNT
         from TAB2
            where
               OPERATION_NO between '50000' and '60000' and
               FINISH_DATIME between timestamp'2004-06-01 20:15:38.00'
                   and timestamp'2004-06-01 20:17:31.00'
                     ) as Q4
           ) as Q3,
         TAB1 as T1
    where
        Q3.C01_ID = T1.C01_ID
        and Q3.C02_LOC = T1.C02_LOC
        and Q3.OPERATION_NO = T1.OPERATION_NO
    ) as Q2,
(select
     ! the query returns the wrong result if Q67 is referenced here
     !
     Q67.C01_ID,          ! Q89.C01_ID,
     Q67.C02_LOC,         ! Q89.C02_LOC,
     Q67.OPERATION_NO,    ! Q89.OPERATION_NO,
    Q67.C07_COUNT       ! Q89.C07_COUNT
 from
    (select
         C01_ID,
         C02_LOC,
         OPERATION_NO,
         max(C06_COUNT) as C06_COUNT,
         C07_COUNT
     from TAB1
         where
             OPERATION_NO between '50000' and '60000' AND
             FINISH_DATIME between timestamp'2004-06-01 20:15:38.00'
               and timestamp'2004-06-01 20:17:31.00'
         group by
             C01_ID,
             C02_LOC,
             OPERATION_NO,
             C07_COUNT
           ) as Q67,
        (select
             C01_ID,
             C02_LOC,
             OPERATION_NO,
             max(C06_COUNT) as C06_COUNT,
             C07_COUNT
         from TAB1
             where
                 FINISH_DATIME between timestamp'2004-06-01 20:15:38.00'
                   and timestamp'2004-06-01 20:17:31.00'
             group by
                 C01_ID,
                 C02_LOC,
                 OPERATION_NO,
                 C07_COUNT
           ) as Q89
   where
       Q67.C01_ID = Q89.C01_ID and
       Q67.C02_LOC = Q89.C02_LOC and
       Q67.OPERATION_NO = Q89.OPERATION_NO
   ) as Q5
```

```
where
    Q2.C01_ID = Q5.C01_ID and
    Q2.C02_LOC = Q5.C02_LOC and
    Q2.OPERATION_NO = Q5.OPERATION_NO
;
 Q2.C01_ID     Q2.C02_LOC   Q5.C01_ID
 1K46804417    20           1K46804417
1 row selected
```

The query returns the correct result if the select statement of Q5 is changed to reference Q89 instead of Q67, as in the following example.

```
select
    Q2.C01_ID,
    Q2.C02_LOC,
    Q5.C01_ID
  from
     (select
          Q3.C01_ID,
          Q3.C02_LOC,
         Q3.OPERATION_NO,
          T1.C07_COUNT
      from
         (select
              Q4.C01_ID,
              Q4.C02_LOC,
              Q4.OPERATION_NO,
              Q4.C07_COUNT
          from
             (select
                  C01_ID,
                  C02_LOC,
                  OPERATION_NO,
                  C07_COUNT
              from TAB2
                 where
                  OPERATION_NO between '50000' and '60000' and
                    FINISH_DATIME between timestamp'2004-06-01 20:15:38.00'
                      and timestamp'2004-06-01 20:17:31.00'
                        ) as Q4
               ) as Q3,
              TAB1 as T1
         where
             Q3.C01_ID = T1.C01_ID
             and Q3.C02_LOC = T1.C02_LOC
             and Q3.OPERATION_NO = T1.OPERATION_NO
         ) as Q2,
     (select
          ! the query returns the correct result if Q89 is referenced here
          !
          Q89.C01_ID,
          Q89.C02_LOC,
          Q89.OPERATION_NO,
          Q89.C07_COUNT
      from
         (select
              C01_ID,
              C02_LOC,
              OPERATION_NO,
              max(C06_COUNT) as C06_COUNT,
```

```
            C07_COUNT
          from TAB1
            where
                OPERATION_NO between '50000' and '60000' AND
                FINISH_DATIME between timestamp'2004-06-01 20:15:38.00'
                  and timestamp'2004-06-01 20:17:31.00'
            group by
                C01_ID,
                C02_LOC,
                OPERATION_NO,
                C07_COUNT
              ) as Q67,
             (select
                C01_ID,
                C02_LOC,
                OPERATION_NO,
                max(C06_COUNT) as C06_COUNT,
                C07_COUNT
            from TAB1
                where
                    FINISH_DATIME between timestamp'2004-06-01 20:15:38.00'
                      and timestamp'2004-06-01 20:17:31.00'
                group by
                   C01_ID,
                    C02_LOC,
                    OPERATION_NO,
                    C07_COUNT
              ) as Q89
        where
            Q67.C01_ID = Q89.C01_ID and
            Q67.C02_LOC = Q89.C02_LOC and
            Q67.OPERATION_NO = Q89.OPERATION_NO
        ) as Q5
where
    Q2.C01_ID = Q5.C01_ID and
    Q2.C02_LOC = Q5.C02_LOC and
    Q2.OPERATION_NO = Q5.OPERATION_NO
;
 Q2.C01_ID     Q2.C02_LOC   Q5.C01_ID
 1K46804417    20           1K46804417
 1K46804420    20           1K46804420
 1K46804424    20           1K46804424
3 rows selected
```

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

- The query joins four contexts of the same table TAB1.
- The strategy is a cross join with two entries (for example, Q2 and Q5) where the first entry (Q2) is a merge of a match strategy (Q3 and T1) with a sort node, and the second entry (Q5) is another cross join of two entries (Q67 and Q89). For example:

```
Cross block of 2 entries
  Cross block entry 1               <== representing Q2
    Merge of 1 entries
      Merge block entry 1
      Conjunct
      Match
        Outer loop                  <== representing  Q3 and Q4
          Sort                      <== this sort node is another key part
          Merge of 1 entries
```

```
            Merge block entry 1
            Merge of 1 entries
              Merge block entry 1
              Conjunct        Conjunct
             Leaf#01 BgrOnly TAB1 Card=2100000
                BgrNdx1 TAB2_IDX_S3 [1:1] Bool Fan=8
                BgrNdx2 TAB2_IDX_S4 [1:1] Bool Fan=8
        Inner loop      (zig-zag)    <== representing  T1
          Conjunct        Conjunct        Conjunct        Conjunct        Get
          Retrieval by index of relation TAB1
            Index name  TAB1_IDX_S1 [0:0]
    Cross block entry 2                <== representing  Q5
      Merge of 1 entries
        Merge block entry 1
        Cross block of 2 entries
          Cross block entry 1          <== representing  Q89
            Merge of 1 entries
              Merge block entry 1
              Aggregate       Sort    Conjunct        Conjunct
              Leaf#02 BgrOnly TAB1 Card=2100000
               BgrNdx1 TAB1_IDX_S4 [1:1] Fan=8
                BgrNdx2 TAB1_IDX_S3 [2:2] Bool Fan=8
          Cross block entry 2          <== representing  Q67
            Merge of 1 entries
              Merge block entry 1
              Aggregate       Sort    Conjunct        Conjunct
              Leaf#03 BgrOnly TAB1 Card=2100000
                BgrNdx1 TAB1_IDX_S1 [4:4] Bool Fan=9
```

- The subselect queries under the query Q5 contain a GROUP BY clause with MAX aggregate function.
- The index TAB1_IDX_S3 is used to retrieve the outer leg of the cross and also used in the inner leg of the cross.
- Three of the columns referenced by the SELECT statements are defined in TAB1_IDX_S3 as C01_ID, C02_LOC and OPERATION_NO.
- The column FINISH_DATIME referenced by the BETWEEN clause is also defined in TAB1_IDX_S3.
- The SELECT statement of the query Q5 references the columns of the query Q67 which becomes the inner leg of the cross strategy instead of the outer leg as explicitly suggested by the SQL query.

This problem has been corrected in Oracle Rdb Release 7.0.8.

2.1.17 Cursor Fetch Returns Wrong Result in Second Execution

Bugs 4086431, 2882908 and 1329838

If you use cursors to fetch the next record of a list, it is possible that the first fetch after opening a cursor could return the wrong record.

The following code is a simple reproducer of the problem.

```
! the first table contains only 2 rows
!
SELECT SLU_ID,SET_ID,UNDR_SECU FROM TP_INST;
 SLU_ID        SET_ID   UNDR_SECU      INST_TYP
 CBON               2   DE0001135002   BON
 NBON            1007   DE4444444444   BON
```

```
2 rows selected

! the second table also contains 2 rows
!
SELECT ISIN,SET_ID FROM tt_trdd_inst;
 ISIN                SET_ID
 DE0001135002             2
 DE4444444444          1007
2 rows selected

! Declare the host variables and cursor
!
SET TRANSACTION;
DECLARE :SLU_ID CHAR (5);
DECLARE :SET_ID INTEGER;
DECLARE :UNDR_SECU CHAR (12);

DECLARE Cursor_bug CURSOR FOR
SELECT * FROM (
  SELECT T1.SLU_ID, T1.SET_ID, T1.UNDR_SECU  FROM
     TT_TRDD_INST, TP_INST
  WHERE T1.UNDR_SECU = T2.ISIN AND
        T1.SET_ID    = T2.SET_ID AND
        INST_TYP     = 'BON'
  ORDER BY T1.SLU_ID, T1.SET_ID, T1.UNDR_SECU
        ) as TT
WHERE
       (
        ( TT.SLU_ID > :SLU_ID )
                OR
        ( ( TT.SLU_ID = :SLU_ID ) AND
          ( TT.SET_ID > :SET_ID ) )
                OR
        ( ( TT.SLU_ID = :SLU_ID ) AND
          ( TT.SET_ID = :SET_ID ) AND
          ( TT.UNDR_SECU > :UNDR_SECU ) )
                )
commit;

! Define the values of the host variables
!
BEGIN
    SET :SLU_ID = 'CBON';
    SET :SET_ID = 2;
    SET :UNDR_SECU = 'DE0001134997';
END;

! Open the cursor the first time
OPEN Cursor_bug;
! It should fetch the first row 'CBON'
FETCH Cursor_bug;
 SLU_ID        SET_ID   UNDR_SECU
 CBON               2   DE0001135002

commit;

! Repeating the above OPEN/FETCH second time gives wrong result
! 'NBON' instead 'CBON'

SET TRANSACTION;
BEGIN
```

```
    SET :SLU_ID = 'CBON';
    SET :SET_ID = 2;
    SET :UNDR_SECU = 'DE0001134997';
END;

OPEN Cursor_bug;

! It should fetch the first row 'CBON'
FETCH Cursor_bug;
 SLU_ID        SET_ID   UNDR_SECU
 NBON            1007   DE4444444444

commit;
```

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

- The cursor query contains the outer SELECT statement as a derived table wrapped around the inner SELECT statment with two tables.
- The inner SELECT statement contains a WHERE clause with equality predicates joining the two tables and a filter predicate, followed by an ORDER BY clause.
- The outer SELECT statement contains a WHERE clause with two complex OR predicates which reference the columns via the derived table.

When the WHERE clauses within the cursor are defined in a different order, the cursor works fine in all cases.

```
! Swap the first and third expressions under the OR predicates

DECLARE Cursor_bug CURSOR FOR
SELECT * FROM (
  SELECT T1.SLU_ID, T1.SET_ID, T1.UNDR_SECU  FROM
     TT_TRDD_INST, TP_INST
  WHERE T1.UNDR_SECU = T2.ISIN AND
        T1.SET_ID    = T2.SET_ID AND
        INST_TYP     = 'BON'
 ORDER BY T1.SLU_ID, T1.SET_ID, T1.UNDR_SECU
        ) as TT
WHERE
       (
        ( ( TT.SLU_ID = :SLU_ID ) AND
          ( TT.SET_ID = :SET_ID ) AND
          ( TT.UNDR_SECU > :UNDR_SECU ) )
                OR
        ( ( TT.SLU_ID = :SLU_ID ) AND
          ( TT.SET_ID > :SET_ID ) )
                OR
        ( TT.SLU_ID > :SLU_ID )
                )
commit;
```

This problem has been corrected in Oracle Rdb Release 7.0.8.

2.1.18 Query with Two ORDER BY Clauses Returns Wrong Result

Bug 4116362

It was possible to get results in the wrong order if you used a query similar to the following which contains two ORDER BY clauses.

```
SELECT TMP1.LOT_NO, TMP1.CASSETTE_ID, TMP1.TIME_STAMP FROM
    (SELECT CLOT.LOT_NO,
            CAS.CASSETTE_ID,
            HLOT.TIME_STAMP
     FROM FCA_LOT AS CLOT,FHA_LOT AS HLOT,
          FCA_CASSETTE AS CAS ,
          (SELECT LOT_NO, MAX(TIME_STAMP) FROM FHA_LOT GROUP BY LOT_NO)
           AS TMP3 (LOT_NO, TIME_STAMP)
     WHERE (CLOT.LOT_STATUS = 'CMP')  AND
          CLOT.LOT_NO = HLOT.LOT_NO AND
          CLOT.CASSETTE_ID=CAS.CASSETTE_ID AND
          HLOT.LOT_NO = TMP3.LOT_NO AND
          HLOT.TIME_STAMP = TMP3.TIME_STAMP
        ORDER BY HLOT.LOT_NO) AS TMP1
ORDER BY TMP1.LOT_NO;
Tables:
  0 = FCA_LOT
  1 = FHA_LOT
  2 = FCA_CASSETTE
  3 = FHA_LOT
Merge of 1 entries
  Merge block entry 1
  Cross block of 3 entries
    Cross block entry 1
      Conjunct: 0.CASSETTE_ID = 2.CASSETTE_ID
      Match
        Outer loop      (zig-zag)
          Index only retrieval of relation 2:FCA_CASSETTE
            Index name  FCA_CASSETTEI1 [0:0]
       Inner loop      (zig-zag)
          Conjunct: 0.LOT_STATUS = 'CMP'
          Get     Retrieval by index of relation 0:FCA_LOT
            Index name  FCA_LOTI2 [0:0]
    Cross block entry 2
      Index only retrieval of relation 1:FHA_LOT
        Index name  FHA_LOTI1 [1:1]
          Keys: 0.LOT_NO = 1.LOT_NO
    Cross block entry 3
      Conjunct: (1.LOT_NO = 3.LOT_NO) AND (1.TIME_STAMP = <mapped field>)
      Merge of 1 entries
        Merge block entry 1
        Aggregate: 0:MAX (3.TIME_STAMP)
        Index only retrieval of relation 3:FHA_LOT
          Index name  FHA_LOTI1 [1:1]
            Keys: 3.LOT_NO = 0.LOT_NO
 LOT_NO             CASSETTE_ID   TIME_STAMP
 HK1L100152         SH011004      21-DEC-2000 16:23:56.00
 HK4L200099         SH011020       1-JUN-2001 11:36:44.00
 HK1L100343         SH015004      21-DEC-2000 16:34:21.00
 CIMP402051         SK025017       1-DEC-2004 02:45:59.00
 MR6NZ6N027         SM091027      18-NOV-2003 10:43:21.00
5 rows selected
```

The query works if the outermost ORDER BY clause is removed.

```
SELECT TMP1.LOT_NO, TMP1.CASSETTE_ID, TMP1.TIME_STAMP FROM
    (SELECT CLOT.LOT_NO,
            CAS.CASSETTE_ID,
           HLOT.TIME_STAMP
     FROM FCA_LOT AS CLOT,FHA_LOT AS HLOT,
          FCA_CASSETTE AS CAS ,
          (SELECT LOT_NO, MAX(TIME_STAMP) FROM FHA_LOT GROUP BY LOT_NO)
           AS TMP3 (LOT_NO, TIME_STAMP)
     WHERE (CLOT.LOT_STATUS = 'CMP')  AND
          CLOT.LOT_NO = HLOT.LOT_NO AND
          CLOT.CASSETTE_ID=CAS.CASSETTE_ID AND
          HLOT.LOT_NO = TMP3.LOT_NO AND
          HLOT.TIME_STAMP = TMP3.TIME_STAMP
        ORDER BY HLOT.LOT_NO) AS TMP1
!ORDER BY TMP1.LOT_NO                   <== Commented out
;
Tables:
  0 = FCA_LOT
  1 = FHA_LOT
  2 = FCA_CASSETTE
  3 = FHA_LOT
Merge of 1 entries
  Merge block entry 1
  Cross block of 4 entries
    Cross block entry 1
      Conjunct: 0.LOT_STATUS = 'CMP'
      Get     Retrieval by index of relation 0:FCA_LOT
        Index name  FCA_LOTI1 [0:0]
    Cross block entry 2
      Index only retrieval of relation 2:FCA_CASSETTE
        Index name  FCA_CASSETTEI1 [1:1]       Direct lookup
          Keys: 0.CASSETTE_ID = 2.CASSETTE_ID
    Cross block entry 3
      Index only retrieval of relation 1:FHA_LOT
        Index name  FHA_LOTI1 [1:1]
          Keys: 0.LOT_NO = 1.LOT_NO
    Cross block entry 4
      Conjunct: (1.LOT_NO = 3.LOT_NO) AND (1.TIME_STAMP = <mapped field>)
      Merge of 1 entries
        Merge block entry 1
        Aggregate: 0:MAX (3.TIME_STAMP)
        Index only retrieval of relation 3:FHA_LOT
          Index name  FHA_LOTI1 [1:1]
            Keys: 3.LOT_NO = 0.LOT_NO
 LOT_NO             CASSETTE_ID   TIME_STAMP
 CIMP402051         SK025017       1-DEC-2004 02:45:59.00
 HK1L100152         SH011004      21-DEC-2000 16:23:56.00
 HK1L100343         SH015004      21-DEC-2000 16:34:21.00
 HK4L200099         SH011020       1-JUN-2001 11:36:44.00
 MR6NZ6N027         SM091027      18-NOV-2003 10:43:21.00
5 rows selected
```

Notice that the problem query applies the index FCA_CASSETTEI1 with the column 2.CASSETTE_ID at the outermost cross block, while the good query applies the index FCA_LOTI1 with the column 0.LOT_NO, which is the correct one referenced by the ORDER BY clause.

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

- The cursor query contains the outer SELECT statement as a derived table wrapped around the inner

SELECT statment joining three tables and an aggregate query with MAX and GROUP BY clauses.

- The inner SELECT statement contains a WHERE clause with four equality predicates and a filter predicate, followed by an ORDER BY clause, referencing the same column as the GROUP BY clause.
- The outer SELECT statement contains an ORDER BY clause referencing the same column as the inner ORDER BY clause via the derived table.

This problem has been corrected in Oracle Rdb Release 7.0.8.

2.1.19 Query with BETWEEN Clause Slows Down Using Index Full Scan

Bug 3835253

It was possible for a query to run more slowly compared to a previous version of Oracle Rdb. This problem started showing up in Oracle Rdb Release 7.0−71 when the fix for Bug 3144382 was released.

This type of performance problem could occur in a query with a predicate of either a GTR, GEQ, LSS, LEQ, NEQ, CONTAINING or STARTS operator as the leading segment of the index retrieval, as in the following example.

```
select count(*)
 from t_viol_tx v ,
      t_lane l,
      t_plaza p,
      t_agency a
  where
    v.lane_id  = l.lane_id and
    l.plaza_id = p.plaza_id and
    p.agency_id  = a.agency_id
    and plate_state = 'MD' and
    plate_number = '60J181'
    and v.tx_date between date ansi '2003−01−01' and date ansi '2003−01−05'
    and a.is_home_agency = 'T';
3:T_AGENCY
Tables:
  0 = T_VIOL_TX
  1 = T_LANE
  2 = T_PLAZA
  3 = T_AGENCY
Aggregate: 0:COUNT (*)
Conjunct: 0.LANE_ID = 1.LANE_ID
Match
  Outer loop
    Sort: 1.LANE_ID(a)
    Cross block of 3 entries
      Cross block entry 1
        Leaf#01 BgrOnly 3:T_AGENCY Card=21
          Bool: 3.IS_HOME_AGENCY = 'T'
          BgrNdx1 I_AGENCY_SPK [0:0] Fan=21
      Cross block entry 2
      Conjunct: 2.AGENCY_ID = 3.AGENCY_ID
        Get     Retrieval sequentially of relation 2:T_PLAZA
      Cross block entry 3
        Conjunct: 1.PLAZA_ID = 2.PLAZA_ID
        Index only retrieval of relation 1:T_LANE
```

```
          Index name  I_LANE_SD1 [0:0]
  Inner loop      (zig-zag)
    Conjunct: 0.PLATE_STATE = 'MD'
    Conjunct: 0.PLATE_NUMBER = '60J181'
    Conjunct: 0.TX_DATE >= DATE '2003-01-01'
    Conjunct: 0.TX_DATE <= DATE '2003-01-05'
    Get     Retrieval by index of relation 0:T_VIOL_TX
      Index name  I_VIOL_TX_SD2 [0:0]
        Bool: (0.TX_DATE >= DATE '2003-01-01') AND (0.TX_DATE <= DATE
              '2003-01-05')

           0
1 row selected
```

The query correctly applies a [2:2] index retrieval if the SQL flag 'selectivity' is set.

```
set flags 'selectivity';

! run the same query from above
!
Tables:
  0 = T_VIOL_TX
  1 = T_LANE
  2 = T_PLAZA
  3 = T_AGENCY
Aggregate: 0:COUNT (*)
Cross block of 4 entries
  Cross block entry 1
    Leaf#01 BgrOnly 3:T_AGENCY Card=21
      Bool: 3.IS_HOME_AGENCY = 'T'
      BgrNdx1 I_AGENCY_SPK [0:0] Fan=21
  Cross block entry 2
    Conjunct: 2.AGENCY_ID = 3.AGENCY_ID
    Get     Retrieval sequentially of relation 2:T_PLAZA
  Cross block entry 3
    Conjunct: 1.PLAZA_ID = 2.PLAZA_ID
    Index only retrieval of relation 1:T_LANE
      Index name  I_LANE_SD1 [0:0]
  Cross block entry 4
    Leaf#02 BgrOnly 0:T_VIOL_TX Card=19098133

      Index name  I_LANE_SD1 [0:0]
  Cross block entry 4
    Leaf#02 BgrOnly 0:T_VIOL_TX Card=19098133
      Bool: (0.LANE_ID = 1.LANE_ID) AND (0.PLATE_STATE = 'MD') AND (
            0.PLATE_NUMBER = '60J181') AND (0.TX_DATE >= DATE '2003-01-01') AND
            (0.TX_DATE <= DATE '2003-01-05')
      BgrNdx1 I_VIOL_TX_SD2 [2:2] Fan=10
        Keys: (0.LANE_ID = 1.LANE_ID) AND (0.TX_DATE >= DATE '2003-01-01') AND
              (0.TX_DATE <= DATE '2003-01-05')

           0
1 row selected
```

This problem has been corrected in Oracle Rdb Release 7.0.8.

2.1.20 Left Outer Join View Query with Constant Columns Returns Wrong Result

Bugs 4155086 and 1752645

A query could produce wrong results such as seen in the following example which selects from a left outer join view with constant columns.

```
create view test_vw1 as
select
cast(cast('ABCD' as char(4)) as char(4)) as fld_1,
cast('ABCD' as char(5)) as fld_2,
cast(case when 1=1 then 'YYY' else 'NNN' end as char(3)) as fld_3,
cast('NNN' as char(3)) as fld_4
    from rdb$database t1
    left outer join rdb$database t2 on t2.rdb$file_name = 'asdfasdf'
;
sel  * from test_vw1;
Tables:
  0 = RDB$DATABASE
  1 = RDB$DATABASE
Cross block of 2 entries        (Left Outer Join)
  Cross block entry 1
    Retrieval sequentially of relation 0:RDB$DATABASE
  Cross block entry 2
    Conjunct: 1.RDB$FILE_NAME = 'asdfasdf'
    Get     Retrieval sequentially of relation 1:RDB$DATABASE
 FLD_1   FLD_2   FLD_3   FLD_4
 NULL    NULL    YYY     NULL
1 row selected
```

The result should display the constant columns instead of NULL. Without the view, the query returns the correct result.

```
select
cast(cast('ABCD' as char(4)) as char(4)) as fld_1,
cast('ABCD' as char(5)) as fld_2,
cast(case when 1=1 then 'YYY' else 'NNN' end as char(3)) as fld_3,
cast('NNN' as char(3)) as fld_4
    from rdb$database t1
    left outer join rdb$database t2 on t2.rdb$file_name = 'asdfasdf'
;
Tables:
  0 = RDB$DATABASE
  1 = RDB$DATABASE
Cross block of 2 entries        (Left Outer Join)
  Cross block entry 1
    Retrieval sequentially of relation 0:RDB$DATABASE
  Cross block entry 2
    Conjunct: 1.RDB$FILE_NAME = 'asdfasdf'
    Get     Retrieval sequentially of relation 1:RDB$DATABASE
 FLD_1   FLD_2   FLD_3   FLD_4
 ABCD    ABCD    YYY     NNN
1 row selected
```

This problem was introduced when a resolution was attempted for Bug 1752645.

After that change, the wrong result was produced by a simple query (like the one above) which selects constant columns of nested expressions from a view query with left outer join on two tables.

There is no workaround for this problem.

This problem has been corrected in Oracle Rdb Release 7.0.8.

2.1.21 Bugcheck in COSI_MEM_GET_VM Reading Large Table with a Dynamic Tactic

Bug 3194130

When the dynamic optimizer was chosen for a query and more than 268,435,455 dbkeys were read from a single index scan, the query would fail with an illegal page count parameter exception and bugcheck.

An example of this is shown below.

```
SQL> select * from t1 where f1>1 and f2>1 optimize for total time;
Leaf#01 BgrOnly T1 Card=300000000
  BgrNdx1 I1 [1:0] Fan=308
  BgrNdx2 I2 [1:0] Fan=308
%RDMS-I-BUGCHKDMP, generating bugcheck dump file
MBRADLEY_USR:[BRADLEY]RDSBUGCHK.DMP;
%COSI-F-UNEXPERR, unexpected system error
-SYSTEM-F-ILLPAGCNT, illegal page count parameter
SQL>
```

The bugceck had the following exception and routine calls.

```
***** Exception at 01023834 : COSI_MEM_GET_VM + 000007A4
%COSI-F-UNEXPERR, unexpected system error
-SYSTEM-F-ILLPAGCNT, illegal page count parameter
Saved PC = 00E7F5A4 : RDMS$$SET_EXEBUF_SIZES + 00000164
Saved PC = 00E7AD64 : RDMS$$EXE_LEAF + 00004464
Saved PC = 00E63DF4 : RDMS$$EXE_OPEN + 00000764
```

This problem can be avoided by disabling the dynamic optimizer by either using a query outline with *EXECUTION OPTIONS NONE* or using the *MAX_STABILITY* flag.

This problem has been corrected in Oracle Rdb Release 7.0.8.

2.1.22 Truncating Empty Table Leaves Uncommited Transaction in Journal

Bug 4245771

If an empty table was truncated, an after-image journal (AIJ) entry would be made for the truncate operation but no commit entry would be entered in the journal for the committed transaction. Thus the transaction would appear uncommitted when a recover operation was done with the journal.

For example, the following message could be displayed by an RMU/RECOVER command when recovering a

journal that has missing COMMIT entries:

```
%RMU-I-LOGRECSTAT, transaction with TSN nn:nn is active
```

This message can be safely ignored. To prevent this from occurring, some other kind of update operation would need to be done in the same transaction as the TRUNCATE command. For example, if a row was added to the table prior to the TRUNCATE command then that would prevent this problem from occurring.

This problem has been corrected in Oracle Rdb Release 7.0.8.

2.1.23 Query With GROUP BY and ORDER BY Returned Rows in the Wrong Order

Bug 4239808

The following query with GROUP BY and ORDER BY clauses returned rows in the wrong order. The detailed query strategy appears after the select statement and before the resulting data rows. If you compare the order of the keys in the ORDER BY clause of the SELECT statement with the order of those same keys in the Sort: line of the detailed query strategy, you will see that they are not the same. This pinpoints the cause of the error.

```
set flags 'detail,strategy';
select
   T2.ITYPE,
   T3.UNLY,
   SUM(T1.AMT) as AMOUNT
from
   (select CLRH, SCTRY, SMRKT,
           SIG, SCMD,
           ETYPE, CNO, DSNAME, AMT
    from EDS
    where BID = 1 and
          ETYPE = 1 and
          CNO IN (24,25) ) as T1
join
   EINTR as T2 on
   T2.SCTRY = T1.SCTRY AND
   T2.SMRKT  = T1.SMRKT  AND
   T2.SIG = T1.SIG
join
   ENT_UND as T3 on
   T3.SCMD = T1.SCMD
group by
   T1.CLRH,
   T2.ITYPE,
   T3.UNLY,
   T1.ETYPE,
   T1.CNO,
   T1.DSNAME
order by
   T1.CLRH,
   T1.DSNAME,
   T1.ETYPE,
   T1.CNO,
   T2.ITYPE,
   T3.UNLY;
```

```
 T2.ITYPE   T3.UNLY           AMOUNT
 HSIC       HSI               6.660000000000000E+002
 HSIP       HSI               7.140000000000000E+002
 SFU2       JSE               5.684000000000002E+002
 MHIC       MHI               2.000000000000000E+002
 MHIF       MHI               1.840000000000000E+001
 MHIP       MHI               4.000000000000000E+001

    ... etc. ...

12 rows selected
```

The results appear in the correct order when the query is run under Oracle Rdb Release 7.0.6.2. The value SFU2 in column T2.ITYPE would correctly appear in row 6 (as shown in the example that follows). In the example above, that value is sorted incorrectly and appears in row 3.

As a workaround, the query works if the columns of the GROUP BY clause are rearranged to be in the same order as in the ORDER BY clause.

```
select
   T2.ITYPE,
   T3.UNLY,
   SUM(T1.AMT) as AMOUNT
from
   (select CLRH, SCTRY, SMRKT,
           SIG, SCMD,
           ETYPE, CNO, DSNAME, AMT
    from EDS
    where BID = 1 and
          ETYPE = 1 and
          CNO IN (24,25) ) as T1
join
   EINTR as INST on
   T2.SCTRY = T1.SCTRY AND
   T2.SMRKT  = T1.SMRKT  AND
   T2.SIG = T1.SIG
join
   ENT_UND as UND on
   T3.SCMD = T1.SCMD
group by
   T1.CLRH,
   T1.DSNAME,
   T1.ETYPE,
   T1.CNO,
   T2.ITYPE,
   T3.UNLY
order by
   T1.CLRH,
   T1.DSNAME,
   T1.ETYPE,
   T1.CNO,
   T2.ITYPE,
   T3.UNLY;
 T2.ITYPE   T3.UNLY           AMOUNT
 HSIC       HSI               6.660000000000000E+002
 HSIP       HSI               7.140000000000000E+002
 MHIC       MHI               2.000000000000000E+002
 MHIF       MHI               1.840000000000001E+001
 MHIP       MHI               4.000000000000000E+001
 SFU2       JSE               5.684000000000001E+002
```

```
    ... etc. ...

12 rows selected
```

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

1. The select query contains GROUP BY and ORDER BY clauses where the columns of the ORDER BY clause have a different order from that of the GROUP BY clause.
2. One item of the select list is an aggregate of some sort, for example SUM.
3. The query applies the cross strategy rather than a match strategy.

This problem has been corrected in Oracle Rdb Release 7.0.8.

2.1.24 Query with GROUP BY, ORDER BY Returned Rows in the Wrong Order

Bugs 4239808 and 3197004

A query with a GROUP BY and an ORDER BY clause presented its rows sorted in the wrong order. Below is an example query on the standard Rdb PERSONNEL database. The example first shows the SQL query, followed by a detailed diagnostic output of the Rdb optimizer's query strategy, followed by the rows returned as the result of the query. Certain lines in the example have been annotated with numerals towards the right margin so that reference can be made to those lines in the explanation that follows the example.

```
set flags 'detail,strategy';

select employee_id as emp_id,
       manager_id as mgr_id,
       last_name as last_name
 from employees inner join departments
 on (employee_id = manager_id)                                    1
 group by employee_id, last_name, manager_id                      2
 order by employee_id desc, manager_id desc;                      3
Tables:
  0 = EMPLOYEES
  1 = DEPARTMENTS
Reduce: 0.LAST_NAME, 0.EMPLOYEE_ID, 1.MANAGER_ID
Sort: 0.LAST_NAME(a), 0.EMPLOYEE_ID(d), 1.MANAGER_ID(a)           4
Cross block of 2 entries
  Cross block entry 1
    Get Retrieval sequentially of relation 1:DEPARTMENTS
  Cross block entry 2
    Get Retrieval by index of relation 0:EMPLOYEES
      Index name  EMP_EMPLOYEE_ID [1:1] Direct lookup
        Keys: 0.EMPLOYEE_ID = 1.MANAGER_ID
 EMP_ID   MGR_ID   LAST_NAME
 00374    00374    Andriola
 00207    00207    Babbin
 00205    00205    Bartlett
 00173    00173    Bartlett

 etc.

26 rows selected
```

The query strategy, in this case, showed but a single Sort: line (the line marked 4). The order of the sort keys was different from the order in the query's ORDER BY clause (line 3). The query contains a GROUP BY clause (line 2), which implies that sorting must be done to perform the grouping operation. As a rule, the Rdb optimizer tries to rearrange keys in a GROUP BY sort using the order of the keys in an outer ORDER BY sort. If it can do so, Rdb eliminates one of the two sorts by combining them. In this example, Rdb did so incorrectly, resulting in the wrong order of sort keys in the remaining sort operation of the original two. This explains why the rows were sorted incorrectly.

In the examples, the EMPLOYEE_ID and MANAGER_ID columns are used in an ON clause (line 1) as the condition for joining two tables, EMPLOYEES and DEPARTMENTS. The EMPLOYEE_ID and MANAGER_ID columns appear in the ORDER BY clause (line 3) and those sort keys are considered to be equivalent. One can sort on the one key or sort on the other to achieve the same results. In certain cases (the preceding example showing one of them), the fact that the ORDER BY clause contained two or more equivalent sort keys was one of the conditions to cause the incorrect behavior in Rdb.

The following example shows the corrected behavior.

```
set flags 'detail,strategy';

select employee_id as emp_id,
       manager_id as mgr_id,
       last_name as last_name
 from employees inner join departments
 on (employee_id = manager_id)                                    1
 group by employee_id, last_name, manager_id                      2
 order by employee_id desc, manager_id desc;                      3
Tables:
  0 = EMPLOYEES
  1 = DEPARTMENTS
Sort: 0.EMPLOYEE_ID(d), 1.MANAGER_ID(d)                           5
Reduce: 0.EMPLOYEE_ID, 0.LAST_NAME, 1.MANAGER_ID
Sort: 0.EMPLOYEE_ID(a), 0.LAST_NAME(a), 1.MANAGER_ID(a)           4
Cross block of 2 entries
  Cross block entry 1
    Get Retrieval sequentially of relation 1:DEPARTMENTS
  Cross block entry 2
    Get Retrieval by index of relation 0:EMPLOYEES
      Index name  EMP_EMPLOYEE_ID [1:1] Direct lookup
        Keys: 0.EMPLOYEE_ID = 1.MANAGER_ID
 EMP_ID   MGR_ID   LAST_NAME
 00471    00471    Herbener
 00418    00418    Blount
 00405    00405    Dement
 00374    00374    Andriola

 etc.

26 rows selected
```

This second example, above, shows the query results returned in the desired order. The query strategy now has two sorts performed (one at line 4 for the GROUP BY clause and one at line 5 for the ORDER BY clause).

A third example, below, shows a variation of the previous query. In this next example, the columns in the GROUP BY and the ORDER BY clauses are the same, and the order sorts first in descending order on the first key and in ascending order on the second key. See the line marked 1 in the right margin for the sort order

that Rdb used. Sorting first on MANAGER_ID and then on EMPLOYEE_ID is accceptable because the two columns are equivalent as explained earlier in this note. The problem was that the sorting was done in ascending order.

```
set flags 'detail,strategy';

select employee_id as emp_id,
       manager_id as mgr_id
  from employees inner join departments
  on (employee_id = manager_id)
  group by employee_id, manager_id
  order by employee_id desc, manager_id asc;
Tables:
  0 = EMPLOYEES
  1 = DEPARTMENTS
Reduce: 1.MANAGER_ID, 0.EMPLOYEE_ID
Sort: 1.MANAGER_ID(a), 0.EMPLOYEE_ID(a)                                  1
Cross block of 2 entries
  Cross block entry 1
    Get Retrieval sequentially of relation 1:DEPARTMENTS
  Cross block entry 2
    Index only retrieval of relation 0:EMPLOYEES
      Index name  EMP_EMPLOYEE_ID [1:1] Direct lookup
        Keys: 0.EMPLOYEE_ID = 1.MANAGER_ID
 EMP_ID   MGR_ID
 00164    00164
 00166    00166
 00168    00168
 00173    00173

 ...

26 rows selected
```

The following example shows the corrected behavior. Again, the positions of MANAGER_ID and EMPLOYEE_ID are swapped in the sort order, which is permissible given that the two columns are equivalent in this query. The first key is sorted in descending order and the second key is sorted in ascending order, as they should be.

```
set flags 'detail,strategy';

select employee_id as emp_id,
       manager_id as mgr_id
  from employees inner join departments
  on (employee_id = manager_id)
  group by employee_id, manager_id
  order by employee_id desc, manager_id asc;
Tables:
  0 = EMPLOYEES
  1 = DEPARTMENTS
Reduce: 1.MANAGER_ID, 0.EMPLOYEE_ID
Sort: 1.MANAGER_ID(d), 0.EMPLOYEE_ID(a)                                  1
Cross block of 2 entries
  Cross block entry 1
    Get Retrieval sequentially of relation 1:DEPARTMENTS
  Cross block entry 2
    Index only retrieval of relation 0:EMPLOYEES
      Index name  EMP_EMPLOYEE_ID [1:1] Direct lookup
        Keys: 0.EMPLOYEE_ID = 1.MANAGER_ID
```

```
 EMP_ID   MGR_ID
 00471    00471
 00418    00418
 00405    00405
 00374    00374

 ...

26 rows selected
```

There is no known workaround for this problem.

This problem has been corrected in Oracle Rdb Release 7.0.8.

2.2 SQL Errors Fixed

2.2.1 Unexpected Bugcheck When Using TRACE Statement and Subselect

Bug 3013618

In prior releases of Oracle Rdb, it was possible to receive a bugcheck dump when displaying a subselect with the TRACE statement. For example, the following TRACE statement bugchecks when SET FLAGS 'TRACE' is enabled.

```
SQL> set flags 'trace';
SQL> set noexecute
SQL>
SQL> begin -- block
cont> declare :AGENT_CODE integer = 0;
cont>
cont> trace
cont>     (SELECT CHAIN_CODE
cont>      FROM AGENTS
cont>      WHERE GROUP_CODE IN ('JVH','JV0')
cont>        AND AGENT_CODE = :agent_code
cont>      LIMIT TO 1 ROWS),
cont>     '|';
cont>
cont> end; -- block
%RDMS-I-BUGCHKDMP, generating bugcheck dump file USER2:[TESTING]RDSBUGCHK.DMP;
%SQL-I-BUGCHKDMP, generating bugcheck dump file USER2:[TESTING]SQLBUGCHK.DMP;
%SYSTEM-F-ACCVIO, access violation, reason mask=00,
virtual address=0000000000000000, PC=000000000024A4EC, PS=0000001B
```

A workaround for this problem is to assign the subselect result to a local variable and then use that local variable with the TRACE statement.

This problem has been corrected in Oracle Rdb Release 7.0.8.

2.2.2 Repeated CREATE of a LOCAL TEMPORARY Table Would Bugcheck

Bug 3953460

In prior versions of Oracle Rdb, attempts to roll back a LOCAL TEMPORARY table definition and re-create it would bugcheck. The following example shows this problem:

```
SQL> create local temporary table tt_tmp (f1 integer, f2 char(67))
cont> on commit preserve rows;
SQL> insert into tt_tmp values (12200,'Accounting');
1 row inserted
SQL> rollback;
SQL> create local temporary table tt_tmp (f1 integer, f2 char(67))
cont> on commit preserve rows;
SQL> insert into tt_tmp values (12200,'Accounting');
```

```
%RDMS-I-BUGCHKDMP, generating bugcheck dump file USER2:[TESTING]RDSBUGCHK.DMP;
```

This problem has been corrected in Oracle Rdb Release 7.0.8. Oracle Rdb now correctly removes the old name from the module-specific symbol table.

2.2.3 CREATE VIEW May Fail With a "Deadlock on Client" Error

Bug 4101404

In prior versions of Oracle Rdb, it was possible in rare cases for the CREATE VIEW statement to fail with a DEADLOCK error. This could occur under the following circumstances:

- The database is active with views created and dropped often. Over a long period this will result in the unique identifier for the view growing to the maximum supported by Oracle Rdb.
- Subsequent CREATE VIEW statements must scan the RDB$RELATIONS system table looking for unused identifiers, that is, those that are freed by a DROP VIEW.
- The view being defined references a view or table with COMPUTED BY columns that also include table references (subselects).

The following example shows the problem.

```
SQL> create view DEPT_STATS as
cont>  select DEPARTMENT_NAME, EMPTY_DEPARTMENT
cont>   from DEPARTMENTS;
%RDB-E-DEADLOCK, request failed due to resource deadlock
-RDB-E-NO_META_UPDATE, metadata update failed
-RDMS-F-DEADLOCK, deadlock on client '.....0..' 0000300B0000000400000055
SQL>
```

In this example, the computed column EMPTY_DEPARTMENT references another COMPUTED BY column in DEPARTMENTS that references a different table.

This problem has been corrected in Oracle Rdb Release 7.0.8. Oracle Rdb now correctly handles the inner references to other tables during the locking of the view id.

2.3 RDO and RDML Errors Fixed

2.3.1 Unexpected NOT_LARDY Following LOCK_CONFLICT Exception

Bug 2274845

In prior releases of Oracle Rdb, when you used the RDO or RDBPRE interfaces you could receive a RDMS−F−NOT_LARDY error after a RDB−F−LOCK_CONFLICT occurred. This error would repeat for all accesses to the table which caused the LOCK_CONFLICT error. A ROLLBACK or COMMIT was required to clear this error state.

The following example shows the problem:

```
INVOKE DATABASE FILENAME TEST_DATABAS

START_TRANSACTION READ_WRITE CONCURRENCY NOWAIT

FOR CLI IN CLI_DNI WITH CLI.DNI_CLI = 'DNI37'
  PRINT CLI.COD_IDI,CLI.FE_ULT_CON
  MODIFY CLI USING
    CLI.COD_IDI='CAS'
  END_MODIFY
END_FOR
%RDB-E-LOCK_CONFLICT, request failed due to locked resource
-RDMS-F-LCKCNFLCT, lock conflict on logical area 57

FOR CLI IN CLI_DNI WITH CLI.DNI_CLI = 'DNI37'
  PRINT CLI.COD_IDI,CLI.FE_ULT_CON
  MODIFY CLI
    USING CLI.COD_IDI='CAS'
  END_MODIFY
END_FOR
%RDMS-F-NOT_LARDY, area for 57:1257:4 not in proper ready mode
```

This problem has been corrected in Oracle Rdb Release 7.0.8.

2.4 Oracle RMU Errors Fixed

2.4.1 RMU Backup Did Not Always Create an RMUBUGCHK.DMP File for Certain Errors

Bug 3668497

For unexpected fatal errors that occurred at certain phases during an Oracle Rdb database backup, the error was displayed but an RMUBUGCHK.DMP file was not created. These errors included SS$_ACCVIO and similar unexpected statuses, including COSI$_ACCVIO, COSI$_BUGCHECK and COSI$_UNEXPERR. This problem happened because a fatal error status which should cause an RMUBUGCHK.DMP file to be created was not passed to the error exception handler. This problem has been fixed and now an RMUBUGCHK.DMP file will be created in these cases.

The following example shows that if the unexpected SS$_ACCVIO VMS system access violation was signaled during a backup of a database, an RMUBUGCHK.DMP file was not always created by Oracle RMU. No message was displayed by Oracle RMU that referenced the file specification of the RMUBUGCHK.DMP file.

```
$RMU/BACKUP MF_PERSONNEL MFP.RBF
%SYSTEM-F-ACCVIO, access violation, reason mask=00, virtual
address=0000000000CC2000, PC=00000000003C4290, PS=0000001B
%RMU-F-FATALERR, fatal error on BACKUP
%RMU-F-FTL_BCK, Fatal  error for BACKUP operation at  1-JUN-2004 11:41:31.23
```

This problem has been corrected in Oracle Rdb Release 7.0.8.

2.4.2 RMU Extract May Bugcheck if Executed When Metadata Changes Are Being Made

Bug 3684674

In previous releases of Oracle Rdb, it was possible that the RMU Extract process could bugcheck if other sessions were modifying the Rdb system table via commands such as DROP or ALTER. For example:

```
$ RMU/EXTRACT/OUT=DSM DSM
%RDMS-I-BUGCHKDMP, generating bugcheck dump file DUMMY:[DUMMY]RDSBUGCHK.DMP;
%RDB-F-BUG_CHECK, internal consistency check failed
%RMU-F-FATALRDB, Fatal error while accessing Oracle Rdb.
%RMU-F-FTL_RMU, Fatal error for RMU operation at 10-JUN-2004 14:20:58.06
```

The bugcheck summary is:

- Alpha OpenVMS 7.2-1
- Oracle Rdb Server 7.0.7.1
- Got a RDSBUGCHK.DMP
- COSI-F-BUGCHECK, internal consistency failure
- Exception occurred at DIOBND$FETCH_AIP_ENT + 000001D4
- Called from DIOBND$GET_LACB + 00000144

- Called from RDMS$TOP_DATABASE_INFO + 00001D24
- Called from BLI$CALLG + 000000BC
- Running image RMUEXTRACT70.EXE

This problem occurs because the DROP operation has marked as deleted the logical area used by a table, storage map, or index. The delete of the logical area happens immediately and is visible therefore to both read−only and read−write users. When the RMU Extract process requests information for the now deleted logical area, a bugcheck occurs noting the internal consistency problem.

In this release of Oracle Rdb, the bugcheck has been replaced with a more informative message as shown below:

```
SQL> create database filename dsm
cont> create storage area rdb$system filename dsm
cont> create storage area a1 filename a1;
SQL> create table t1(i1 integer);
SQL> create storage map t1m for t1 store in a1;
SQL> commit;
SQL>  $ RMU/EXTRACT/OUTPUT=DSM DSM
SQL> drop storage map t1m;
SQL>  $ RMU/EXTRACT/OUTPUT=DSM DSM
%RDB−E−OBSOLETE_METADA, request references metadata objects that no longer exist
−RDMS−F−CANTFINDLAREA, cannot locate logical area 47 in area inventory page list
%RMU−F−FATALRDB, Fatal error while accessing Oracle Rdb.
%RMU−F−FTL_RMU, Fatal error for RMU operation at 10−JUN−2004 11:46:52.36
SQL>
```

This problem has been corrected in Oracle Rdb Release 7.0.8. If this error occurs, repeat the RMU Extract command as this will start a new transaction.

2.4.3 RMU Restore/Incremental/Area Did Not Check if Default and List Areas Were Out of Date

Bug 3676698

On an incremental by area restore where the specified list of storage areas to be restored does not include RDB$SYSTEM, a check is done to see if the backup RBF file contains the RDB$SYSTEM area. If it does and the incremental backup TSN of the live RDB$SYSTEM area is less than the TSN of the RDB$SYSTEM area in the RBF backup file (the live RDB$SYSTEM area is out of date), Oracle RMU requires that the newer RDB$SYSTEM area in the backup RBF file be added to the list of areas to be restored. In this case, Oracle RMU returns the error message:

```
%RMU−F−RDBSYSTEMREQ, The RDB$SYSTEM storage area must also be specified
```

This is to prevent database corruption.

However, if a LIST (SEGMENT) or DEFAULT storage area other than RDB$SYSTEM were defined for the database, no check was made for these areas. These areas are now checked for being out of date on an incremental by area RMU database restore the same way that the RDB$SYSTEM storage area is checked. The RDBSYSTEMREQ message has been changed to include one, two, or all three of these areas as necessary.

```
%RMU−F−RDBSYSTEMREQ, The RDB$SYSTEM MF_PERS_DEFAULT MF_PERS_SEGSTR storage
```

```
area(s) must also be specified for the restore.
```

The following example shows that previously only an out of data RDB$SYSTEM storage area was checked for on an incremental by area restore.

```
$rmu/restore/incremental/noconfirm/nocdd/nolog/area jobs.rbf JOBS
%RMU-F-RDBSYSTEMREQ, The RDB$SYSTEM storage area must also be specified
%RMU-F-FTL_RSTR, Fatal error for RESTORE operation at 14-JUL-2004 16:17:49.33
```

The following example shows that now the default and list (segment) areas will be checked to make sure they are not out of date in addition to the RDB$SYSTEM storage area on an incremental by area restore.

```
$rmu/restore/incremental/noconfirm/nocdd/nolog/area jobs.rbf JOBS
%RMU-F-RDBSYSTEMREQ, The RDB$SYSTEM MF_PERS_DEFAULT MF_PERS_SEGSTR storage
 area(s) must also be specified for the restore.
%RMU-F-FTL_RSTR, Fatal error for RESTORE operation at 14-JUL-2004 16:17:49.33
```

This problem has been corrected in Oracle Rdb Release 7.0.8.

2.4.4 RMU Verify Access Violation When Readying an Invalid Logical Area

Bug 3776327

An access violation occurred in the RMU Verify process if a bad logical area ID number was detected because of database corruption. The corruption occurred when the RMU Verify process was readying logical areas while verifying database data pages. The RMU Verify process correctly reported the bad logical area ID, but when the process attempted to continue the database verification, it failed to check if a bad storage area ID number had been generated based on the bad logical area ID. The bad storage area ID was then used to get the address of the storage area entry in the database root. This caused the access violation and has been fixed.

The following example shows the problem:

```
$RMU/VERIFY/ALL/LOG BAD_DB

%RMU-I-BGNSEGPAG, beginning verification of RDB$SYSTEM storage area
%RMU-W-CANTFINDLAREA, cannot locate logical area 312 in area inventory page list
%RMU-E-BDLAREADY, error readying logical area with dbid 312
%SYSTEM-W-ACCVIO, access violation, reason mask=00, virtual address=000000000000
000B, PC=00000000003BB3E8, PS=0000001B

%RMU-F-ABORTVER,  fatal error encountered; aborting verification
%SYSTEM-F-ACCVIO, access violation, reason mask=00, virtual address=000000000000
000B, PC=00000000003BB3E8, PS=0000001B
%RMU-F-FATALOSI, Fatal error from the Operating System Interface.
%RMU-I-BUGCHKDMP, generating bugcheck dump file SERDB_USER1:[HOCHULI]RMUBUGCHK
.DMP;
```

This problem has been corrected in Oracle Rdb Release 7.0.8.

2.4.5 RMU Backup Access Violation When Backing Up the Root ACL

Bug 3866383

An access violation could occur during the RMU Backup process of an Oracle Rdb database when the database root Access Control List was being backed up. This happened because of a problem testing for the end of the memory allocated to hold the ACL. This problem did not happen often. It only occurred when certain memory boundary conditions were reached.

The following example shows the access violation:

```
$rmu/backup/online/log rdms_db bckdir:rdms_db.rbf
 %RMU-I-QUIETPT, waiting for database quiet point at 16-AUG-2004 02:45:10.65
 %RMU-I-RELQUIETPT, Database quiet point lock has been released at
        16-AUG-2004 02:45:10.81.
 %SYSTEM-F-ACCVIO, access violation, reason mask=00, virtual
        address=0000000000CF2000, PC=00000000003E9620, PS=0000001B
  %RMU-F-FATALERR,  fatal error on BACKUP
  %RMU-F-FTL_BCK, Fatal error for BACKUP operation at 16-AUG-2004 02:45:10.91
```

This problem can be avoided by using the NOACL qualifier on the backup command and recreating the database root ACL after the restore.

This problem has been corrected in Oracle Rdb Release 7.0.8.

2.4.6 RMU Load/Parallel Command Could Hang if it was Repeated

Bug 3803694

The RMU Load/Parallel command could hang if it was repeated one or more times. This happened because an event flag used to coordinate communication between the root process and the executor processes was not properly initialized.

The following example shows the parallel load hanging when the load was repeated.

```
$ rmu/unload  [.db]testdb  tab1  tab1.unl
%RMU-I-DATRECUNL,  6 data records unloaded.

$ rmu/load/log/parallel=(buf=50,exec=2) -    ! normal successful completion
    [.db]testdb  tab1  tab1.unl
%RMU-I-EXECUTORMAP, Executor EXECUTOR_1 (pid: 2F03372D) will load storage
area AREA1.
%RMU-I-EXECUTORMAP, Executor EXECUTOR_2 (pid: 2F03292E) will load storage
area AREA2.
EXECUTOR_2:  %RMU-I-DATRECSTO,  3 data records stored.
EXECUTOR_1:  %RMU-I-DATRECSTO,  3 data records stored.

$ rmu/load/log/parallel=(buf=50,exec=2) -    ! hangs forever
    [.db]testdb  tab1  tab1.unl
%RMU-I-EXECUTORMAP, Executor EXECUTOR_1 (pid: 2F033130) will load storage
area AREA1.
```

```
%RMU-I-EXECUTORMAP, Executor EXECUTOR_2 (pid: 2F031731) will load storage
area AREA2.
```

This problem has been corrected in Oracle Rdb Release 7.0.8.

2.4.7 Incorrect Backup of Empty Single AIJ File

Bug 3878051

Starting with Oracle Rdb Release 7.0.7.3, if you backed up an empty single extendable AIJ file, that is, an AIJ with only an open record, the "last commit TSN" field in the AIJ file open record was incorrectly reset to zero as shown below:

```
$ rmu/dump/after/nodata foo_aij.aij

1/1             TYPE=O, LENGTH=510, TAD=24-SEP-2004 04:48:43.32, CSM=00
    Database $111$DUA4:[VIGIER.RECO_BUG.DB]FOO.RDB;2
    Database timestamp is 24-SEP-2004 04:48:41.11
    Facility is "RDMSAIJ ", Version is 711.1
    Database version is 71.0
    AIJ Sequence Number is 0
    Last Commit TSN is 0:0
   .
   .
   .
```

If you tried to recover the AIJ file, the recovery process ignored all transactions from the file:

```
$ rmu/recover foo_aij.aij
%RMU-I-LOGRECDB, recovering database file $111$DUA4:[VIGIER.RECO_BUG.DB]
FOO.RDB;1
%RMU-I-LOGOPNAIJ, opened journal file RAID1:[VIGIER.RECO_BUG.DB]FOO_AIJ.AIJ;1
at 24-SEP-2004 05:07:56.35
%RMU-I-LOGRECSTAT, transaction with TSN 0:128 ignored
%RMU-I-AIJONEDONE, AIJ file sequence 0 roll-forward operations completed
%RMU-I-LOGRECOVR,  0 transactions committed
%RMU-I-LOGRECOVR,  0 transactions rolled back
%RMU-I-LOGRECOVR,  1 transaction ignored
   .
   .
   .
```

To avoid the problem, you can use multiple circular AIJs instead of a single extendable AIJ.

When you back up an empty single extendable AIJ file, no backup file is produced, and the AIJ open record is left unchanged. The AIJ roll forward now works as expected.

This problem has been corrected in Oracle Rdb Release 7.0.8.

2.4.8 RMU Collect Statistics=Storage Caused Incorrect Row Clustering Factors

Bug 4047055

The RMU Collect Optimizer Statistics=Storage command could produce incorrect row clustering factor values if cardinality statistics were not also collected. If storage statistics were the only statistics collected, the table cardinality values needed to calculate the row clustering factors were added to the previous table cardinality values. The table cardinality values should have been cleared before they were recalculated.

The following example shows a case where collecting only storage statistics created row clustering factors which were half the correct value which was returned if both storage and cardinality values were collected. The correct row clustering factors would also have been produced if the Statistics qualifier was not specified because the default is Statistics=(Cardinality,Workload,Storage). The difference is due to the cardinality being in effect doubled since it was added to the previous cardinality which had not changed. Cardinality is used as a divisor in calculating the row cluster factor which caused the row cluster factors to be half the correct value if only Storage statistics were collected.

```
$ rmu/collect optimizer mf_personnel /statistics=(storage)/log=opt_stor.log
$ rmu/collect optimizer mf_personnel /statistics=(storage,cardinality)-
_$ /log=opt_storcard.log
$ search opt*.log "row clustering"

*****************************

    DEVICE:[DIRECTORY]OPT_STORCARD.LOG;2

      Row clustering factor  : 0.0112994

      Row clustering factor  : 0.0114943

      Row clustering factor  : 1.0000000

      Row clustering factor  : 0.4285714

      Row clustering factor  : 0.0089286

      Row clustering factor  : 0.0180723

      Row clustering factor  : 0.0361446

      Row clustering factor  : 0.0131579

      Row clustering factor  : 0.0087464

      Row clustering factor  : 0.0000000

    *****************************

    DEVICE:[DIRECTORY]OPT_STOR.LOG;2

      Row clustering factor  : 0.0056497

      Row clustering factor  : 0.0057143

      Row clustering factor  : 0.5000000

      Row clustering factor  : 0.2142857

      Row clustering factor  : 0.0044643

      Row clustering factor  : 0.0090361

      Row clustering factor  : 0.0180723
```

```
      Row clustering factor  : 0.0065789

      Row clustering factor  : 0.0043732

      Row clustering factor  : 0.0000000
```

To avoid this problem, do not specify the collection of only storage statistics but also collect cardinality statistics. Do any of the following:

```
$ rmu/collect optimizer mf_personnel /statistics=(storage,cardinality)
$ rmu/collect optimizer mf_personnel −
_$ /statistics=(storage,cardinality,workload)
$ rmu/collect optimizer mf_personnel
```

This problem has been corrected in Oracle Rdb Release 7.0.8.

2.4.9 Unexpected ACCVIO and BUGCHECK from RMU Extract

Bug 4205822

In prior versions of Oracle Rdb, attempts to use RMU Extract to format a view, trigger or constraint definition that contained a dbkey literal (for example, _DBKEY'−1:−1:−1') would fail with an error similar to that shown below:

```
$ rmu/extract/item=view/option=(filename_only,noheader,match=myv%) sql$database
set verify;
set language ENGLISH;
set default date format 'SQL92';
set quoting rules 'SQL92';
set date format DATE 001, TIME 001;
attach 'filename MF_PERSONNEL';
create view MYV
    (JRN,
     FMUB) as
%SYSTEM−F−ACCVIO, access violation, reason mask=00,
virtual address=0000000000000000, PC=FFFFFFFF81096B64, PS=0000001B
%RMU−F−FATALOSI, Fatal error from the Operating System Interface.
%RMU−I−BUGCHKDMP, generating bugcheck dump file USER2:[TESTER]RMUEXTBUGCHK.DMP;
%RMU−F−FTL_RMU, Fatal error for RMU operation at 26−FEB−2005 03:31:09.78
```

This problem has been corrected in Oracle Rdb Release 7.0.8. The following example shows the corrected output from RMU Extract. Please note that SQL converts _ROWID into _DBKEY literals so that this is what will be extracted by RMU.

```
$ rmu/extract/item=view/option=(filename_only,noheader,match=myv%) sql$database
set verify;
set language ENGLISH;
set default date format 'SQL92';
set quoting rules 'SQL92';
set date format DATE 001, TIME 001;
attach 'filename MF_PERSONNEL';
create view MYV
    (JRN,
     FMUB) as
    (select
        case C1.DBKEY
```

```
            when _DBKEY'-1:-1:-1' then 'D'
            else 'I'
        end,
        C1.MY_UB
    from MYONE C1);

commit work;
```

2.4.10 RMU Extract Fails on Some View Definitions

Bug 1267126
Bug 1729528
Bug 2130670
Bug 3294003
Bug 3418796
Bug 3518990
Bug 3810178

In prior releases of Rdb, RMU Extract failed to correctly extract view definitions when the view contained nested UNION operators, derived tables or other complexities. Incorrect syntax would be generated for these cases.

This release also corrects a memory management problem which may cause RMU Extract to bugcheck.

These problems have been corrected in Oracle Rdb Release 7.0.8.

Chapter 3
Software Errors Fixed in Oracle Rdb Release 7.0.7.3

This chapter describes software errors that are fixed by Oracle Rdb Release 7.0.7.3.

3.1 Software Errors Fixed That Apply to All Interfaces

3.1.1 COSI−F−INVCLADTY During Timestamp Conversion

Bug 3544859

When attempting a data type conversion involving a timestamp data type, a COSI−F−INVCLADTY error could occur.

The following example shows this.

```
select * from timetable where departure_time > timestamp
    '2004−02−02 00:00:00.00'

Caused:
%RDB−E−CONVERT_ERROR, invalid or unsupported data conversion
−COSI−F−INVCLADTY, invalid class data type combination in descriptor

Where:
departure_time is TIMESTAMP(2)
```

A possible workaround for the above example might be the following.

```
select * from timetable where cast(departure_time as date vms) >
    date vms ' 2−FEB−2004 00:00:00.00'
```

This problem has been corrected in Oracle Rdb Release 7.0.7.3.

3.1.2 Wrong Order on Descending and Ascending Sort on Same Column

Bug 3197004

When a query contains ORDER BY and GROUP BY clauses, the Rdb Optimizer attempts to eliminate sorts where possible. One way it does so is by rearranging the order of keys in a GROUP BY clause to match the order in some ORDER BY clause and then it only performs a single sort instead of two sorts.

The following sample query contains multiple GROUP BY and ORDER BY clauses.

```
select  t1.name, t2.name, t1.datum, t2.datum
   from (select name, datum from a
            group by name, datum) t1
         join
         (select name, datum from b
            group by name, datum) t2
         on (t1.name = t2.name and t1.datum = t2.datum)
    group by t1.name, t2.name, t1.datum, t2.datum
    order by t1.name desc, t2.name asc, t1.datum desc, t2.datum asc;
```

Note that the final ORDER BY clause sorts first on the name column in descending order and then on the name column in ascending order. Before the problem was corrected, the Rdb Optimizer incorrectly produced

a query strategy that sorted everything in ascending order.

There is no known workaround for this problem.

This problem has been corrected in Oracle Rdb Release 7.0.7.3.

3.1.3 Bugcheck at RDMS$FIND_CJOIN + 0000B69C

Bug 3395635

A query worked without incident by itself. However, after a query outline was created and the query was re−executed, a bugcheck in Rdb was reported during query compilation. The bugcheck dump file showed that an access violation had occurred at location RDMS$$FIND_CJOIN + 0000B69C.

The problem was reproducible using the Rdb sample database MF_PERSONNEL. Below are the commands entered to show the problem.

```
create index emp_last_name on employees (last_name);

create outline sv_pp
id '4A7CF117DBBA752D84D109077954A59E'
mode 0
as (
  query (
  -- For loop
   subquery (
     subquery (
       subquery (
          EMPLOYEES 1    access path index      EMP_LAST_NAME
              join by cross to
          JOB_HISTORY 0  access path index      JOB_HISTORY_HASH
          )
          union with
        subquery (
          CANDIDATES 2    access path sequential
          )
        )
      )
    )
   )
 compliance optional;

select cj.last_name, cj.first_name, cj.employee_id, cj.job_code,
       cj.department_code, cj.supervisor_id, cj.job_start
  from (select e.last_name, e.first_name, e.employee_id,
               jh.job_code, jh.department_code,
               jh.supervisor_id, jh.job_start
          from job_history jh, employees e
          where jh.employee_id = e.employee_id) cj
       left outer join
       (select last_name, first_name
          from candidates
          order by last_name, first_name) c
       on c.last_name = cj.last_name
       where cj.last_name is null;
```

There is no known workaround for this problem.

This problem has been corrected in Oracle Rdb Release 7.0.7.3.

3.1.4 JOIN Query with a Function in Predicate Bugchecks

Bug 3499717

The following INNER JOIN query with a function applied to one of the equi−join predicates bugchecks.

```
select count(*)
from
  (select xxxx_id, yyyy_id
     from  TAB1 where
          operation_no = '50300' and
          yyyy_loc between '00' and '99'
  ) as q1
inner join
  (select xxxx_id, product_id
     from  TAB2 where
          operation_no = '01300'
  ) as q2
on q1.xxxx_id = q2.xxxx_id and
   trim(q1.yyyy_id) = q2.xxxx_id ;
RDMS−I−BUGCHKDMP, generating bugcheck dump file DISK:[DIRECTORY]RDSBUGCHK.DMP;
***** Exception at 00FE8E18 : RDMS$$FIND_VALID_SEG_CRTV + 000000B8
```

As a workaround, the query works if the SQL flag 'NOMAX_SOLUTION' is set, as in the following example.

```
SQL> set flags 'nomax_solution';
  OR
$DEFINE RDMS$$DISABLE_MAX_SOLUTION 1

Here is the new strategy of the new run:

Tables:
  0 = TAB1
  1 = TAB2
Aggregate: 0:COUNT (*)
Cross block of 2 entries
  Cross block entry 1
    Merge of 1 entries
      Merge block entry 1
      Conjunct: 1.OPERATION_NO = '01300'
      Get     Retrieval sequentially of relation 1:TAB2
  Cross block entry 2
    Merge of 1 entries
      Merge block entry 1
      Conjunct: (0.XXXX_ID = 1.XXXX_ID) AND (TRIM (BOTH ' ' FROM 0.YYYY_ID) =
                1.XXXX_ID)
      Leaf#01 BgrOnly 0:TAB1 Card=10
        Bool: (0.OPERATION_NO = '50300') AND (0.YYYY_LOC >= '00') AND (
              0.YYYY_LOC <= '99')
        BgrNdx1 TAB1_IDX_S1 [2:2] Fan=9
          Keys: (0.XXXX_ID = 1.XXXX_ID) AND (0.YYYY_LOC >= '00') AND (0.YYYY_LOC
                <= '99')
          Bool: 0.OPERATION_NO = '50300'

           0
1 row selected
```

This problem is caused by the fix made for Bug 1635351 in Oracle Rdb Release 7.0.6.3.

This problem has been corrected in Oracle Rdb Release 7.0.7.3.

3.1.5 DIOBND$FETCH_AIP_ENT + 000001D4 Bugcheck

Bug 3360543

READ ONLY transactions sometimes caused a bugcheck in DIOBND$FETCH_AIP_ENT when the metadata required by the transaction was being manipulated by another process.

The bugcheck has been replaced by the following error message:

```
%RDB-E-OBSOLETE_METADA, request references metadata objects that no longer exist
-RDMS-F-CANTFINDLAREA, cannot locate logical area nnn in area inventory page
list
```

This problem has been corrected in Oracle Rdb Release 7.0.7.3. The problem was that the READ ONLY transaction had knowledge of a now obsolete table or index logical area. Attempts to use that index or table partition now fail. When this problem occurs, the transaction should ROLLBACK and start a new transaction that will read the latest metadata from the database.

This problem has been corrected in Oracle Rdb Release 7.0.7.3.

3.1.6 Various Bugchecks and Corruptions With Indexes of Type is Sorted Ranked

Bugs 3424257, 3289081, 3364208, 3424296 and 3013421

While adding or removing records in an index of *TYPE IS SORTED RANKED*, it was possible that one of several different bugchecks or corruptions could be observed.

Bugchecks could occur that do not usually indicate index corruption including various bugchecks in the following routines:

- PSII2SPLITNODE
- PSII2REMOVEDUPBBC
- PSII2INSERTDUPBBC

Bugchecks could occur that usually do indicate index corruption including bugchecks in the following routines:

- PSII2REMOVET
- PSIINDEX2GETSEPAFTER

In most cases, the corruption of the index would be reported by *RMU/VERIFY* as either an inconsistant cardinality or an invalid or negative amount of free space in a node.

The following example shows the types of errors that may be reported by *RMU/VERIFY* for a corrupt index.

```
%RMU-W-BTRLENERR, area RDB$SYSTEM, page 68, line 5
                  b-tree node length error
                  expected node length 58, found: 0
%RMU-I-BTRERPATH, parent B-tree node of 66:68:5 is at 66:18:3
...
%RMU-I-BTRENTLEN, B-tree node entry 1 has an invalid data length of 3345.
%RMU-I-BTRNODDBK, Dbkey of B-tree node is 210:5307:1
...
%RMU-W-BTRLEACAR, Inconsistent leaf cardinality (C2) of 1 specified
                   for entry 29 at dbkey 210:6196:0 using precision of 33.
```

Under rare circumstances, it was possible that when multiple records were deleted from the same key value in the same transaction, a *%RDB-E-NO_RECORD* error could be returned. A subsequent attempt to delete the same record would succeed without error. In this case, the dbkey being reported would be the dbkey of a duplicates overflow node in the index affected by the delete.

The following example shows the type of message returned. The dbkey is a dbkey of an index overflow node that collapsed and was deleted in a previous operation within the same transaction.

```
%RDB-E-NO_RECORD, access by dbkey failed because dbkey is no longer associated
with a record
-RDMS-F-NODBK, 66:71:0 does not point to a data record
```

If any index corruptions are detected they can be removed by dropping and recreating the offending index.

As a workaround, the problem can be avoided by using alternate index types.

This problem has been corrected in Oracle Rdb Release 7.0.7.3. Once this version is installed, it is strongly recommended that all indices of *TYPE IS SORTED RANKED* be verified using a command similar to *RMU/VERIFY/INDEX/DATA*. If any corruptions are reported, the index must be dropped and recreated to remove the corruption.

3.1.7 Query with OR Predicate Slows Down Due to Wrong Strategy

Bug 3319289

The following query suffers in performance when the optimizer does not apply static OR index retrieval strategy.

```
set flags 'strategy,detail';
select count(*) from ( SELECT col33, col37 FROM MYTABLE1
            WHERE col34 = -463362512350317888 and
                ( ('A' = 'A' and
                    col10 >= '20031001' and
                    col10 <= '20031202' ) OR
                  ('B' = 'C' and
                    col09 >= '20031001' and
                    col09 <= '20031202' ) ) AND
                col20 >= 0 AND
                col20 <= 9999999999999 AND
                col13 >= ' ' AND
                col13 <= 'Z';
Tables:
  0 = MYTABLE1
```

```
Aggregate: 0:COUNT (*)
Merge of 1 entries
  Merge block entry 1
  Leaf#01 BgrOnly 0:MYTABLE1 Card=8483
    Bool: (0.COL34 = -463362512350317888) AND ((('A' = 'A') AND (0.COL10 >=
          '20031001') AND (0.COL10 <= '20031202')) OR (('B' = 'C') AND (0.COL09
           >= '20031001') AND (0.COL09 <= '20031202'))) AND (0.COL20 >= 0) AND (
          0.COL20 <= 9999999999999) AND (0.COL13 >= ' ') AND (0.COL13 <= 'Z')
    BgrNdx1 X2_MYTABLE [1:1] Fan=9
      Keys: 0.COL34 = -463362512350317888
    BgrNdx2 X7_MYTABLE [0:0] Fan=7
      Bool: (0.COL20 >= 0) AND (0.COL20 <= 9999999999999)

        8483
1 row selected
```

The same query runs much faster in Oracle Rdb Release 7.0.6.2, picking the right indices with static OR index retrieval.

```
Tables:
  0 = MYTABLE1
Aggregate: 0:COUNT (*)
Merge of 1 entries
  Merge block entry 1
  Conjunct: (0.COL34 = -463362512350317888) AND ((('A' = 'A') AND (0.COL10 >=
            '20031001') AND (0.COL10 <= '20031202')) OR (('B' = 'C') AND (
            0.COL09 >= '20031001') AND (0.COL09 <= '20031202'))) AND (0.COL20
             >= 0) AND (0.COL20 <= 9999999999999) AND (0.COL13 >= ' ') AND (
            0.COL13 <= 'Z')
  OR index retrieval
    Conjunct: ('A' = 'A') AND (0.COL10 >= '20031001') AND (0.COL10 <= '20031202'
              )
    Get     Retrieval by index of relation 0:MYTABLE1
      Index name  X2_MYTABLE [2:2]
        Keys: (0.COL34 = -463362512350317888) AND (0.COL10 >= '20031001') AND (
              0.COL10 <= '20031202')
        Bool: 'A' = 'A'
    Conjunct: NOT (('A' = 'A') AND (0.COL10 >= '20031001') AND (0.COL10 <=
              '20031202')) AND ('B' = 'C') AND (0.COL09 >= '20031001') AND (
              0.COL09 <= '20031202')
    Get     Retrieval by index of relation 0:MYTABLE1
      Index name  X3_MYTABLE [2:2]
        Keys: (0.COL34 = -463362512350317888) AND (0.COL09 >= '20031001') AND (
              0.COL09 <= '20031202')
        Bool: 'B' = 'C'
```

Notice that Oracle Rdb Release 7.0.6.2 applies static OR index retrieval while Oracle Rdb Release 7.0.7 uses regular index retrieval strategy.

As a workaround, if the index X1_MYTABLE is removed, the query works like it did in Release 7.0.6.2 and it still works if the same index is created again.

This problem has been corrected in Oracle Rdb Release 7.0.7.3.

3.2 SQL Errors Fixed

3.2.1 Corrupt Storage Map Possible if String Literals Incorrectly Used for Numeric Columns

Bug 3530235

In prior releases of Oracle Rdb, it was permitted to create a storage map for an index or a table having the WITH LIMIT OF as character strings even though the column type was numeric. However, the resulting storage map was corrupted and would not allow rows to be inserted.

The following example shows this problem.

```
SQL> create database filename testdb
cont>    create storage area sa1;
SQL> create table t1 (col1 integer);
SQL> create storage map t1_map for t1
cont>   store using (col1)
cont>       in sa1 with limit of ('10');
%SQL-I-NUMCMPTXT, Numeric column will be compared with string literal as text
SQL> commit;
SQL> insert into t1 values (1);
%RDB-E-CONVERT_ERROR, invalid or unsupported data conversion
-RDMS-E-CSETBADASSIGN, incompatible character sets prohibit the requested
assignment
```

This problem may be observed when upgrading from an older Rdb release using SQL IMPORT. In this case, the solution is to include the redefinition of the STORAGE MAP in the IMPORT command so that the WITH LIMIT OF values are numeric.

This problem has been corrected in Oracle Rdb Release 7.0.7.3. The procedure which upgraded the character literals to numeric during CREATE STORAGE MAP and CREATE INDEX has been corrected to support literals with character set data.

3.2.2 Unexpected SQL-F-QUETOOBIG Error During CREATE TABLE

Bug 3542644

In prior versions of Oracle Rdb, a CREATE TABLE or ALTER TABLE statement with many constraints, DEFAULT, AUTOMATIC AS, or COMPUTED BY definitions could fail with the following error.

```
%SQL-F-QUETOOBIG, Query or routine contains too many table references
```

A workaround for this problem is to split the CREATE TABLE or ALTER TABLE into small ALTER TABLE statements.

This problem has been corrected in Oracle Rdb Release 7.0.7.3. SQL now resets the counter for each constraint, default value and COMPUTED BY expression allowing many more definitions in a single

command.

3.2.3 Performance and Limits Problems with Concatenation Operator

Bugs 2641023 and 3539378

In prior versions of Oracle Rdb, the CONCAT or || operator could cause excessive resource usage when used in a query under the ORACLE LEVEL1 or ORACLE LEVEL2 dialects, such as through SQL*Net for Rdb. Reported problems included:

- Query fails with BUFLIMEXC errors

```
%RDB-E-IMP_EXC, facility-specific limit exceeded
-RDMS-F-BUFLIMEXC, internal buffer limit exceeded
```

- Query incurs excessive page faults during execution

These problems were caused by overly complex query generation required to implement Oracle semantics for NULL and zero length string usage for the concatenation operator. Oracle RDBMS semantics require that concatenation treat NULL as a zero length string, and conversely that VARCHAR function results (TRIM, SUBSTRING, RTRIM, TO_CHAR, etc) have zero length treated identically with NULL.

These problems have been corrected in Oracle Rdb Release 7.0.7.3. When concatenation is mixed with VARCHAR string functions or other concatenation operators, further optimizations are performed to minimize the complexity of the generated query. Note that this problem did not occur with any other dialect.

3.3 Oracle RMU Errors Fixed

3.3.1 Aborted AIJ Backup may Cause Database Shutdown via DBR Failure

Bug 1712408

In previous releases of Oracle Rdb, when aborting a quiet point AIJ backup of a single extensible journal, it was possible that the database would be shutdown after DBR failure. The DBR log file, if enabled, ended with the following:

```
<timestamp> − After−image journaling is being shutdown with hard data loss
AIJ shutdown reason: journal has been possibly deleted or moved
```

When doing a quiet point backup of a single extensible journal, the journal will be truncated and reinitialized. During that phase, the file did not look like an AIJ file to Rdb. So, if the backup was aborted during that phase, then the DBR would not find the AIJ file and would abort, shutting down the database.

Now the DBR detects such a condition and so it will truncate and reinitialize the AIJ file to continue the recovery. The DBR log file, if enabled, will show:

```
<timestamp> − Recovering AIJ information
<timestamp> − Recovering AIJ backup which failed at initialization time
<timestamp> − AIJ file has been reinitialized
<timestamp> − Initializing AIJ EOF to 1:2
```

To avoid such a situation, you should use multiple circular AIJs.

This problem has been corrected in Oracle Rdb Release 7.0.7.3.

3.3.2 RMU/ANALYZE/TRANSACTION_TYPE=READ_ONLY Could Hold Quiet Point Lock in CR Mode

Bug 3456640

The position of the /TRANSACTION_TYPE qualifier during an RMU/ANALYZE command caused the quiet point lock to be held in CR mode for the following case:

```
$RMU/ANALYZE/INDEX/OPTION=DEBUG  SQL$DATABASE  /TRANSACTION=READ_ONLY
```

However, the quiet point lock was held in NL mode for the following case:

```
$RMU/ANALYZE/INDEX/OPTION=DEBUG/TRANSACTION=READ_ONLY  SQL$DATABASE
```

This could cause delay in an RMU/BACKUP/ONLINE which waits to acquire the quiet point lock. Now, in both these cases, the quiet point lock will be correctly held in NL mode.

The following example shows that if the /TRANSACTION_TYPE=READ_ONLY qualifier was placed after the database name parameter instead of before the database name parameter in an RMU/ANALYZE

command, the quiet point lock was held in CR mode during the RMU/ANALYZE operation instead of in the correct NL mode.

In the first process:

```
$RMU/ANALYZE/INDEX/OPTION=DEBUG  SQL$DATABASE  /TRANSACTION=READ_ONLY
```

In another process, display the locks for the first process while the RMU/ANALYZE executes.

```
$RMU/SHOW LOCKS/PROCESS=2060FEEE

================================================================================
SHOW LOCKS/PROCESS Information
================================================================================
--------------------------------------------------------------------------------
Resource: quiet

          ProcessID Process Name        Lock ID   System ID Requested Granted
          --------- ---------------     --------- --------- --------- -------
Owner:    2060FEEE  Proc First 1..      54004849  00010003            CR
```

The workaround for this problem is to place the /TRANSACTION_TYPE=READ_ONLY qualifier before the database parameter.

In the first process:

```
$RMU/ANALYZE/INDEX/OPTION=DEBUG/TRANSACTION=READ_ONLY  SQL$DATABASE
```

In another process, display the locks for the first process while the RMU/ANALYZE executes.

```
$RMU/SHOW LOCKS/PROCESS=2060FEEE

================================================================================
SHOW LOCKS/PROCESS Information
================================================================================
--------------------------------------------------------------------------------
Resource: quiet

          ProcessID Process Name        Lock ID   System ID Requested Granted
          --------- ---------------     --------- --------- --------- -------
Owner:    2060FEEE  Proc First 1..      54004849  00010003            NL
```

This problem has been corrected in Oracle Rdb Release 7.0.7.3.

3.4 LogMiner Errors Fixed

3.4.1 RMU /UNLOAD /AFTER_JOURNAL Transaction Commit Timestamp Accuracy Increase

In prior versions of Oracle Rdb, the RMU /UNLOAD /AFTER_JOURNAL command derived the transaction commit timestamp from the after−image journal "group commit" time. Multiple after−image journal records can exist within a single group and all share the same timestamp. Usually, this presents no difficulty. The transaction commit time extracted by the Oracle Rdb LogMiner(tm) was "close enough" to the actual time that the transaction committed.

However, in some situations, the transaction commit time reported by LogMiner could be earlier than the transaction start time. This condition appears most often in a heavily loaded system with very short duration transactions when the AIJ Log Server (ALS) feature was enabled.

In such a configuration, it was possible for a transaction to start and complete while the ALS process was collecting information to write to the after−image journal. Because the ALS process uses a single time stamp for all groups in an I/O, the user−captured transaction start time could be later than the ALS captured group time. And because the RMU /UNLOAD /AFTER_JOURNAL command returns the transaction commit time from the "group commit" time, it could appear that the transaction committed before it started.

This situation has been changed. The transaction commit time is now explicitly captured by each transaction during the commit sequence and written to the after−image journal. The RMU /UNLOAD /AFTER_JOURNAL command extracts this time value as the actual transaction commit time.

AIJ Content Change

Because the LogMiner(tm) now extracts the transaction commit time from the journal in a different way, journals created with an earlier version of Rdb but whose transaction records are extracted with this release will claim an unpredictable and incorrect value for the commit time. If the accuracy of the transaction commit time is important to you, use the prior version of Oracle Rdb to unload after−image journals created with that version.

3.4.2 RMU /UNLOAD /AFTER_JOURNAL Possible Missing Pre−delete Content

Bug 3569886

In very rare cases, it was possible for the RMU /UNLOAD /AFTER_JOURNAL command to omit deleted record contents from an extract operation. The LogMiner was incorrectly processing the after−image journal when a record's pre−delete content was followed in the journal by a physical record delete indicator for the same physical DBKEY.

This problem has been corrected in Oracle Rdb Release 7.0.7.3. The RMU /UNLOAD /AFTER_JOURNAL command now uses a modified sort comparison phase that correctly handles the unexpected order in the after−image journal so that the pre−delete record content is extracted as expected.

Chapter 4
Software Errors Fixed in Oracle Rdb Release 7.0.7.2

This chapter describes software errors that are fixed by Oracle Rdb Release 7.0.7.2.

4.1 Software Errors Fixed That Apply to All Interfaces

4.1.1 Wrong Index Retrieval is Selected in Query with Range List Predicates

Bug 3243452

The following query slows down significantly when the range list index retrieval [0:1,1:0] is wrongly applied instead of [3:3] for greater and less than predicates.

```
set flags 'strategy,detail';
select count(*)
  from product_division pd
        inner join product_warehouse pw on
                pw.company_no = pd.company_no and pw.division_no =
                 pd.division_no and pw.product_no = pd.product_no
        left outer join wm_location wml on
                wml.company_no = pd.company_no
                and wml.division_no = pd.division_no
                and wml.warehouse_no = pw.warehouse_no
                and wml.slot_id = pw.slot_id
where pd.company_no = 1
and pw.division_no = 1
and pd.product_no not between 700000 and 799999
and pd.product_no < 900000
and pd.product_status_cd not in ('D','O','R');
Tables:
  0 = PRODUCT_DIVISION
  1 = PRODUCT_WAREHOUSE
  2 = WM_LOCATION
Aggregate: 0:COUNT (*)
Conjunct: (0.PRODUCT_NO < 700000) OR (0.PRODUCT_NO > 799999)
Cross block of 2 entries        (Left Outer Join)
  Cross block entry 1
    Cross block of 2 entries
      Cross block entry 1
        Conjunct: 0.COMPANY_NO = 1
        Conjunct: (0.PRODUCT_NO < 700000) OR (0.PRODUCT_NO > 799999)
        Conjunct: 0.PRODUCT_NO < 900000
        Conjunct: (0.PRODUCT_STATUS_CD <> 'D') AND (0.PRODUCT_STATUS_CD <> 'O')
        Conjunct: 0.PRODUCT_STATUS_CD <> 'R'
        Leaf#01 NdxOnly 0:PRODUCT_DIVISION Card=153560
          Bool: 0.DIVISION_NO = 1
          FgrNdx  PRODUCT_DIVISION_CMP_NDX [2:2] Fan=33
            Keys: (0.DIVISION_NO = 1) AND (0.COMPANY_NO = 1)
          BgrNdx1 PRODUCT_DIVISION_PRD_NDX [0:1] Fan=33
            Keys: 0.PRODUCT_NO < 900000
      Cross block entry 2
        Conjunct: 1.DIVISION_NO = 1
        Leaf#02 BgrOnly 1:PRODUCT_WAREHOUSE Card=153560
          Bool: (1.COMPANY_NO = 0.COMPANY_NO) AND (1.DIVISION_NO = 0.DIVISION_NO
                ) AND (1.PRODUCT_NO = 0.PRODUCT_NO)
          BgrNdx1 PRODUCT_WAREHOUSE_PRIMARY [0:1,1:0] Fan=33 <= Should be [3:3]
            Keys: r0: 1.PRODUCT_NO > 799999
                  r1: 1.PRODUCT_NO < 700000
            Bool: (1.PRODUCT_NO < 900000) AND (1.COMPANY_NO = 1) AND (
```

```
                1.DIVISION_NO = 1)
  Cross block entry 2
    Conjunct: (2.COMPANY_NO = 0.COMPANY_NO) AND (2.DIVISION_NO = 0.DIVISION_NO)
               AND (2.WAREHOUSE_NO = 1.WAREHOUSE_NO) AND (2.SLOT_ID = 1.SLOT_ID)
    Index only retrieval of relation 2:WM_LOCATION
      Index name  WM_LOCATION_PRMRY [4:4]
        Keys: (2.SLOT_ID = 1.SLOT_ID) AND (2.WAREHOUSE_NO = 1.WAREHOUSE_NO) AND
              (2.DIVISION_NO = 0.DIVISION_NO) AND (2.COMPANY_NO = 0.COMPANY_NO)

          0
1 row selected
```

The query applies [3:3] index retrieval if the SQL flag 'SELECTIVITY' is enabled.

The strategy output is similar to the above except for the following lines pointed to by <= in the inner Cross block entry 2.

```
      Cross block entry 2
        Conjunct: 1.DIVISION_NO = 1
        Leaf#02 BgrOnly 1:PRODUCT_WAREHOUSE Card=153560
          Bool: (1.COMPANY_NO = 0.COMPANY_NO) AND (1.DIVISION_NO = 0.DIVISION_NO
                ) AND (1.PRODUCT_NO = 0.PRODUCT_NO)
          BgrNdx1 PRODUCT_WAREHOUSE_PRIMARY [3:3] Fan=33             <=
            Keys: (1.DIVISION_NO = 0.DIVISION_NO) AND (1.PRODUCT_NO  <=
                  0.PRODUCT_NO) AND (1.COMPANY_NO = 0.COMPANY_NO)    <=
            Bool: (1.PRODUCT_NO < 900000) AND ((1.PRODUCT_NO < 700000) OR ( <=
                  1.PRODUCT_NO > 799999)) AND (1.COMPANY_NO = 1) AND (      <=
                  1.DIVISION_NO = 1)
```

This is a regression caused by the fix for Bug 2634849 in Oracle Rdb Release 7.1.2.

This problem has been corrected in Oracle Rdb Release 7.0.7.2.

4.1.2 Applications That Use $HIBER/$WAKE Hang in HIB

Bug 2881846

User applications that utilize the OpenVMS $HIBER/$WAKE system services to process asynchronous events could hang in "HIB" state if the database had the FAST COMMIT feature enabled. The hung process would resume executing normally if a $WAKE was issued against it by another process.

For example, the Oracle SQL/Services product utilizes $HIBER/$WAKE to coordinate events between the server processes (dispatcher and executor). The SQL/Services executor processes could sometimes hang in "HIB" state.

This problem would sometimes occur when a global checkpoint request was issued. A global checkpoint will occur whenever Oracle Rdb switches to another journal, or when a checkpoint is manually requested by issuing an RMU/CHECKPOINT command. A race condition between the database checkpoint activity and the application's usage of $WAKE could cause a $WAKE intended for the application to be consumed by the database checkpoint activity, preventing the application from properly waking from its hibernate state.

This problem can be avoided by disabling the fast commit feature. Note that this can have a significant impact on performance.

This problem has been corrected in Oracle Rdb Release 7.0.7.2.

4.1.3 Bugchecks at PIOABW$SYNCH_PAGE + 00000564

Bug 3264272

If a process did not have sufficient quota it was possible to encounter bugchecks like the following:

```
***** Exception at 0109556C : PIOABW$SYNCH_PAGE + 00000564
%RDMS-F-CANTWRITEDBS, error writing pages 2:367-369
-SYSTEM-F-EXQUOTA, process quota exceeded
```

While in general it is necessary to have sufficient quotas to support the total number of potential concurrent disk I/Os possible, certain operations, like asynchronous batch−writes, can safely ignore these errors. If an asynchronous batch−write encounters an EXQUOTA error, the disk write can be retried later when another attempt is made to write the buffer. The asynchronous batch−write operations have been modified to tolerate EXQUOTA errors.

Note that there are many other operations that may still fail with a bugcheck if an EXQUOTA error is encountered when attempting to read or write database disk files. For example, EXQUOTA errors encountered when attempting I/O to the recovery−unit journal (.RUJ) or after−image journal (.AIJ) will still result in a fatal bugcheck. Also, if an attempt to write out all modified buffers is unable to start any I/Os successfully, then a bugcheck will still occur.

This problem has been corrected in Oracle Rdb Release 7.0.7.2.

4.1.4 Query Bugcheck After Logical RDMS$SET_FLAGS Set to SELECTIVITY(2)

Bug 3242615

Before the problem was fixed, a query resulted in a bugcheck when the query was executed after the sampled selectivity capability was enabled. Sampled selectivity can be enabled in any of several ways, one of which is to define the VMS logical name RDMS$SET_FLAGS to be SELECTIVITY(2). Below is an example of a failing query:

```
select count (*) from xxx
  where (c in (-1, 0) and b > 48955) or
        (a > 30000 and b > 49990) or
        (a > 49900 and b > 20000);
```

The problem occurs when there is an OR condition in the WHERE clause of the query.

As a workaround, enable sampled selectivity only for specific queries rather than for all queries within an application. This can be done by using the OPTIMIZE WITH SAMPLED SELECTIVITY clause on specific queries rather than defining RDMS$SET_FLAGS as SELECTIVITY(2).

This problem has been corrected in Oracle Rdb Release 7.0.7.2.

4.1.5 Bugchecks or Corruption With Indexes of Type is Sorted Ranked

Bug 3009262

When updating an index of *TYPE IS SORTED RANKED*, it was possible that a bugcheck or index corruption could occur. This problem could occur if the index being updated allowed duplicates and was most likely when there existed a large number of duplicates for a particular key value and the duplicate rows were widely distributed in the database.

When a key value was changed or a row was deleted requiring the removal of a dbkey from a duplicates chain, it was possible that a bugcheck dump could be generated.

The following example shows an update that causes a key value to change. During the attempt to remove the dbkey for the affected row from the appropriate duplicates chain, Rdb generates a bugcheck dump.

```
SQL> update tt11 set f1 = 'b' where f1 = 'a' and f3 = 5 and f2 < 125;
%RDMS-I-BUGCHKDMP, generating bugcheck dump file
USER1:[BUGCHECK_DIR]RDSBUGCHK.DMP;
%COSI-F-BUGCHECK, internal consistency failure
```

The bugcheck could contain either of the two exceptions shown in the following example.

```
***** Exception at 00A32E98 : PSII2SCANGETNEXTBBCDUPLICATE + 00000120
%COSI-F-BUGCHECK, internal consistency failure

***** Exception at 00C53E4C : PSII2REMOVEDUPBBC + 0000130C
%COSI-F-BUGCHECK, internal consistency failure
```

Under rare circumstances, it was possible that the insertion of a dbkey into a duplicates chain could cause the duplicates chain to become corrupt. The corruption could be removed by dropping and recreating the offending index.

As a workaround, the problem can be avoided by using alternate index types, or by adding columns to the index to make it more unique.

This problem has been corrected in Oracle Rdb Release 7.0.7.2.

4.1.6 Query with Constant Column in UNION/GROUP BY Returns Wrong Results

Bug 3336416

The following query containing a constant column in one of the UNION legs with a GROUP BY clause returns the wrong result.

```
set flags 'strategy,detail';

SELECT CTEGEN
FROM
 (SELECT CODSOC,CTEGEN,CDEV,SUM(SIGNE) AS SIGNE
    FROM (
```

```
        (SELECT CODSOC,CTEGEN,CDEV,SUM(MONTANT_SIGNE) AS SIGNE
         FROM T1
         WHERE
            CTEGEN NOT LIKE '7%' AND  PERIODE = '2001'
         GROUP BY CODSOC,CTEGEN,CDEV)
      UNION
        (SELECT CODSOC,'59100000',CDEV,SUM(MONTANT_SIGNE) AS SIGNE
         FROM T1
         WHERE
            CTEGEN LIKE '7%' AND  PERIODE = '2001'
         GROUP BY CODSOC,CTEGEN,CDEV)
         ) AS BBB
    GROUP BY CODSOC,CTEGEN,CDEV) as
    BBB(CODSOC,CTEGEN,CDEV,SIGNE)
WHERE
    SIGNE=0 AND
    CODSOC = 'MUT' AND CTEGEN LIKE '121%';
Tables:
  0 = T1
  1 = T1
Conjunct: <mapped field> = 0
Conjunct: <mapped field> LIKE '121%'
Merge of 1 entries
  Merge block entry 1
  Aggregate: 0:SUM (<mapped field>)
  Sort: <mapped field>(a), <mapped field>(a), <mapped field>(a)
  Conjunct: '59100000' LIKE '121%'              <== See Note
  Merge of 1 entries
    Merge block entry 1
    Reduce: <mapped field>, <mapped field>, <mapped field>, <mapped field>
    Sort: <mapped field>(a), <mapped field>(a), <mapped field>(a),
          <mapped field>(a)
    Merge of 2 entries
      Merge block entry 1
      Aggregate: 1:SUM (0.MONTANT_SIGNE)
      Conjunct: (0.CODSOC = 'MUT') AND (0.CTEGEN LIKE '121%')
      Conjunct: NOT (0.CTEGEN LIKE '7%') AND (0.PERIODE = '2001')
      Get     Retrieval by index of relation 0:T1
        Index name  T1_NDX [3:3]
          Keys: (0.PERIODE = '2001') AND (<mapped field> = 'MUT') AND (
                <mapped field> LIKE '121%')
          Bool: (0.CTEGEN LIKE '121%') AND (NOT (0.CTEGEN LIKE '7%'))
      Merge block entry 2
      Aggregate: 2:SUM (1.MONTANT_SIGNE)
      Conjunct: 1.CODSOC = 'MUT'
      Conjunct: '59100000' LIKE '121%'
      Conjunct: (1.CTEGEN LIKE '7%') AND (1.PERIODE = '2001')
      Get     Retrieval by index of relation 1:T1
        Index name  T1_NDX [3:3]
          Keys: (1.PERIODE = '2001') AND (<mapped field> = 'MUT') AND (1.CTEGEN
                 LIKE '7%')
          Bool: ('59100000' LIKE '121%') AND (1.CTEGEN LIKE '7%')
0 rows selected
```

Note:: The conjunct containing the constants in both operands is incorrectly placed outside of the merge (UNION query).

This problem occurs when the query contains the following:

1. The main table is derived from a subquery of UNION legs.

2. Each UNION leg contains a GROUP BY clause.
3. The subquery SELECT statement contains an aggregate function SUM on a column which is NOT part of the GROUP BY columns.
4. One of the UNION legs contains a constant value as the SELECT column.
5. One of the WHERE predicates of the main query references the column that is mapped to the constant value column of the UNION leg.

As a workaround, the query works if the constant column is wrapped inside a CASE statement, as in the following example.

```
SELECT CTEGEN
FROM
 (SELECT CODSOC,CTEGEN,CDEV,SUM(SIGNE) AS SIGNE
    FROM (
        (SELECT CODSOC,CTEGEN,CDEV,SUM(MONTANT_SIGNE) AS SIGNE
         FROM T1
         WHERE
            CTEGEN NOT LIKE '7%' AND  PERIODE = '2001'
         GROUP BY CODSOC,CTEGEN,CDEV)
      UNION
        (SELECT CODSOC,
!           '59100000',         ! <== wrap this constant in CASE statement
            CASE CTEGEN WHEN NULL THEN '********' ELSE '59100000' END
            CDEV,SUM(MONTANT_SIGNE) AS SIGNE
            CDEV,
          SUM(MONTANT_SIGNE) AS SIGNE
        FROM T1
        WHERE
            CTEGEN LIKE '7%' AND  PERIODE = '2001'
        GROUP BY CODSOC,CTEGEN,CDEV)
        ) AS BBB
    GROUP BY CODSOC,CTEGEN,CDEV) as
    BBB(CODSOC,CTEGEN,CDEV,SIGNE)
WHERE
    SIGNE=0 AND
    CODSOC = 'MUT' AND CTEGEN LIKE '121%';
Tables:
  0 = T1
  1 = T1
Conjunct: <mapped field> = 0
Merge of 1 entries
  Merge block entry 1
  Aggregate: 0:SUM (<mapped field>)
  Sort: <mapped field>(a), <mapped field>(a), <mapped field>(a)
  Merge of 1 entries
    Merge block entry 1
    Reduce: <mapped field>, <mapped field>, <mapped field>, <mapped field>
    Sort: <mapped field>(a), <mapped field>(a), <mapped field>(a),
          <mapped field>(a)
    Conjunct: 0.CTEGEN LIKE '121%'      <== See Note1
    Merge of 2 entries
      Merge block entry 1
      Aggregate: 1:SUM (0.MONTANT_SIGNE)
      Conjunct: (0.CODSOC = 'MUT') AND (0.CTEGEN LIKE '121%')
      Conjunct: NOT (0.CTEGEN LIKE '7%') AND (0.PERIODE = '2001')
      Get     Retrieval by index of relation 0:T1
        Index name  T1_NDX [3:3]
          Keys: (0.PERIODE = '2001') AND (<mapped field> = 'MUT') AND (
                <mapped field> LIKE '121%')
```

```
          Bool: (0.CTEGEN LIKE '121%') AND (NOT (0.CTEGEN LIKE '7%'))
      Merge block entry 2
      Aggregate: 2:SUM (1.MONTANT_SIGNE)
      Conjunct: 1.CODSOC = 'MUT'
      Conjunct: (1.CTEGEN LIKE '7%') AND (1.PERIODE = '2001')
      Get     Retrieval by index of relation 1:T1
        Index name  T1_NDX [3:3]
          Keys: (1.PERIODE = '2001') AND (<mapped field> = 'MUT') AND (1.CTEGEN
                 LIKE '7%')
          Bool: 1.CTEGEN LIKE '7%'
 CTEGEN
 12111090
1 row selected
```

Note1:: The correct conjunct is now generated instead of the wrong one.

This problem has been corrected in Oracle Rdb Release 7.0.7.2.

4.1.7 Left Outer Join Query with Sub−select Overflows the Stack

Bugs 3357593 and 2649215

The following left outer join query with a sub−select overflows the stack.

```
SELECT
    (select ct.name from
      (select firm_id, cust_id, addr_id,
        (select name from t1  where
            t2.addr_id = t1.addr_id) as name
                 FROM t2) as ct
      where  ct.firm_id = t5.firm_id and
             ct.cust_id = t5.cust_id) as cust_name
FROM
    stock t3
    left outer join t4
        on t4.firm_id   = t3.firm_id and
           t4.trade_id  = t3.trade_id and
           t4.trade_pos = t3.trade_pos
    left outer join t5
        on t5.firm_id   = t4.firm_id and
           t5.trade_id  = t4.trade_id;
%RDB−F−IMP_EXC, facility−specific limit exceeded
−RDMS−F−XPR_STACK_OFLO, expression forces too many levels of recursion
```

The problem occurs when the query selects from three tables (t3, t4 and t5) of left outer join and contains a sub−select clause with two tables (t1 and t2) joined by the equality predicates referencing the columns from table t5.

This problem is caused by the fix made for Bug 2649215 which did not cover this particular query.

There is no known workaround for this problem.

This problem has been corrected in Oracle Rdb Release 7.0.7.2.

4.1.8 Signal Failing in Compound Statement

Bug 3364468

A signal in a compound statement could cause SQL/RDB to go into an infinite loop, especially if a label with a "leave label" statement was encountered.

The following example shows this behavior.

```
begin
declare :cnt integer = 0;
declare :tdbkey char(8);

while (exists (select c0 from update_db.module.dp$_2)) loop
        rollback;

        set transaction read write isolation level read committed
        reserving update_db.job_history for shared write;

        label_loop:

        for :dp as each row of cursor dp$_2 for
                select c0 from update_db.module.dp$_2
        do
                set :tdbkey = :dp.c0;

                delete from update_db.module.dp$_2 where current of dp$_2;

                update update_db.job_history set
                        supervisor_id = supervisor_id where dbkey = :tdbkey;

                set :cnt = :cnt + 1;
                if (:cnt >= 30) then leave label_loop;
                end if;
        end for;
        commit;
        if (:cnt >= 1) then
                signal 'JDLAY';
        end if;
end loop;
end;
```

One possible workaround would be to avoid using a "leave label" statement.

This problem has been corrected in Oracle Rdb Release 7.0.7.2.

4.1.9 Bugcheck in KOD$ROLLBACK and PSII2REMOVEDUPBBC with Sorted Ranked Indexes

Bug 3408974

In Oracle Rdb Release 7.0.7.1, a problem was introduced in the handling of sorted ranked indexes by the optimizer.

Depending on the structure and data distribution of index nodes in the sorted ranked index, it was possible that one of the following exceptions would be raised.

```
COSI-F-BUGCHECK, internal consistency failure
Exception occurred at KOD$ROLLBACK + 00000328

COSI-F-BUGCHECK, internal consistency failure
Exception occurred at PSII2REMOVEDUPBBC + 0000142C
```

These problems only occur when a sorted ranked index is being used by the optimizer to filter out possible candidate dbkeys for selection.

A workaround for this problem is to use normal sorted indexes.

This problem has been corrected in Oracle Rdb Release 7.0.7.2.

4.1.10 Database Corruption When LOCKING IS PAGE LEVEL Enabled

Storage areas created using the "LOCKING IS PAGE LEVEL" option would sometimes have missing updates. This problem would occur when there was a lot of contention for pages in the storage area.

For example, if an index update were lost, IDXDATMIS errors may have been displayed by RMU/VERIFY.

```
%RMU-W-IDXDATMIS, Index ALL_DEPT does not point to a row in table
                  DEPARTMENT_REC.
                  Logical dbkey of the missing row is 149:302:5.
```

This problem can be avoided by using the "LOCKING IS ROW LEVEL" option, which is the default.

Note that the "LOCKING IS PAGE LEVEL" option works best with applications that have little contention between users for database pages. Also, ideally, transactions should be short in duration and most if not all pages used by the transaction should be able to simultaneously reside in the buffers allocated to the process.

This problem has been corrected in Oracle Rdb Release 7.0.7.2.

4.1.11 Table Constraint Should Fail on Updating a Row

Bug 3434648

A customer updates a row in the table, changing the value in one column from NULL to some text string. Under Oracle Rdb Release 7.0.6.3, a constraint on that table correctly reports that the update is in error. However in Oracle Rdb Release 7.0.7.1, the constraint failure does not occur.

A simple reproducer using a SELECT statement is created from the UPDATE/COMMIT statement of the original BUG report where the CHECK clause defined in the table constraint contains four OR predicates, such as the following.

```
        predicate-1 OR predicate-2 OR predicate-3 OR predicate-4
```

which is inverted using a NOT operator, such as:

```
   NOT (predicate-1 OR predicate-2 OR predicate-3 OR predicate-4)
```

The query selects "Constraint violation" on the ALIAS_LOC table where the dbkey is the target row that UPDATE performs on. It should find the row that gets the constraint violation when the conditions in the predicates are not met.

```
set flags 'strategy,detail';

Simple reproducer for the problem
---------------------------------

declare :dbk bigint;
select dbkey into :dbk from ALIAS_LOC A limit to 1 row;

select 'Constraint Failure' from ALIAS_LOC A
  where a.dbkey = :dbk and
    NOT (
        ! Part 1
         (A.LOCATION_1_NAME is null and
          A.LOCATION_2_NAME is null and
          A.LOCATION_3_NAME is null)
        OR
        ! Part 2
         ((A.LOCATION_2_NAME is null and        ! <== missing in Cross block 3
           A.LOCATION_3_NAME is null)           ! <== of the strategy
          and exists
          (select * from LOCATION C10
           where
             C10.ORGANIZATION_NAME = A.ORGANIZATION_NAME and
             C10.UNIT_NAME         = A.UNIT_NAME         and
             C10.PARENT_LOCATION_NAME is null               and
             C10.LOCATION_NAME     = A.LOCATION_1_NAME
          )             ! end of select
         )              ! end of or
        OR
        ! Part 3
        ((A.LOCATION_3_NAME is null)
         and exists
         (select * from LOCATION C11, LOCATION C12
           where
               C11.ORGANIZATION_NAME    = A.ORGANIZATION_NAME and
               C11.UNIT_NAME            = A.UNIT_NAME         and
               C11.PARENT_LOCATION_NAME is null                  and
               C11.LOCATION_NAME        = A.LOCATION_1_NAME   and
               C12.PARENT_LOCATION_NAME = C11.LOCATION_NAME   and
               C12.LOCATION_NAME        = A.LOCATION_2_NAME
          )             ! end of select
         )              ! end of or
        OR
        ! Part 4
        (exists
         (select * from LOCATION C13, LOCATION C14, LOCATION C15
          where
               C13.ORGANIZATION_NAME    = A.ORGANIZATION_NAME and
               C13.UNIT_NAME            = A.UNIT_NAME         and
               C13.PARENT_LOCATION_NAME is null                  and
               C13.LOCATION_NAME        = A.LOCATION_1_NAME   and
               C14.PARENT_LOCATION_NAME = C13.LOCATION_NAME   and
               C14.LOCATION_NAME        = A.LOCATION_2_NAME   and
               C15.PARENT_LOCATION_NAME = C14.LOCATION_NAME   and
```

```
              C15.LOCATION_NAME        = A.LOCATION_3_NAME
         )              ! end of select
        )               ! end of or
      )
  limit to 1 row;
Tables:
  0 = ALIAS_LOC
  1 = LOCATION
  2 = LOCATION
  3 = LOCATION
  4 = LOCATION
  5 = LOCATION
  6 = LOCATION
Firstn: 1
Cross block of 4 entries
  Cross block entry 1
    Conjunct: NOT MISSING (0.LOC_NAME_1) OR NOT MISSING (0.LOC_NAME_2) OR NOT
              MISSING (0.LOC_NAME_3)
    Conjunct: 0.DBKEY = <var0>
    Firstn: 1
    Get     Retrieval by DBK of relation 0:ALIAS_LOC
  Cross block entry 2
    Conjunct: NOT MISSING (0.LOC_NAME_3) OR (<agg0> = 0)
    Aggregate-F1: 0:COUNT-ANY (<subselect>)
    Cross block of 2 entries
      Cross block entry 1
        Index only retrieval of relation 2:LOCATION
          Index name  LOCATION_NDX [4:4]         Direct lookup
            Keys: (2.ORG_NAME = 0.ORG_NAME) AND (2.UNIT_NAME = 0.UNIT_NAME) AND
                  (MISSING (2.P_LOC_NAME)) AND (2.LOC_NAME = 0.LOC_NAME_1)
      Cross block entry 2
        Conjunct: (3.P_LOC_NAME = 2.LOC_NAME) AND (3.LOC_NAME = 0.LOC_NAME_2)
        Index only retrieval of relation 3:LOCATION
          Index name  LOCATION_NDX [0:0]
  Cross block entry 3
    Conjunct: <agg1> = 0        ! <=== See Note below.
    Aggregate-F1: 1:COUNT-ANY (<subselect>)
    Index only retrieval of relation 1:LOCATION
      Index name  LOCATION_NDX [4:4]         Direct lookup
        Keys: (1.ORG_NAME = 0.ORG_NAME) AND (1.UNIT_NAME = 0.UNIT_NAME) AND (
              MISSING (1.P_LOC_NAME)) AND (1.LOC_NAME = 0.LOC_NAME_1)
  Cross block entry 4
    Conjunct: <agg2> = 0
    Aggregate-F1: 2:COUNT-ANY (<subselect>)
    Cross block of 3 entries
      Cross block entry 1
        Index only retrieval of relation 4:LOCATION
          Index name  LOCATION_NDX [4:4]         Direct lookup
            Keys: (4.ORG_NAME = 0.ORG_NAME) AND (4.UNIT_NAME = 0.UNIT_NAME) AND
                  (MISSING (4.P_LOC_NAME)) AND (4.LOC_NAME = 0.LOC_NAME_1)
      Cross block entry 2
        Conjunct: (5.P_LOC_NAME = 4.LOC_NAME) AND (5.LOC_NAME = 0.LOC_NAME_2)
        Index only retrieval of relation 5:LOCATION
          Index name  LOCATION_NDX [0:0]
      Cross block entry 3
        Conjunct: (6.P_LOC_NAME = 5.LOC_NAME) AND (6.LOC_NAME = 0.LOC_NAME_3)
        Index only retrieval of relation 6:LOCATION
          Index name  LOCATION_NDX [0:0]
0 rows selected
```

NOTE:: Note that the following conjunct under the Cross block entry 3 should contain the following predicates, but that is missing.

```
    Conjunct: NOT MISSING (0.LOC_NAME_2) OR NOT MISSING (0.LOC_NAME_3) OR
                (<agg1> = 0)
```

This problem was caused by the fix for Bug 2285818 where the query contains predicates shared by other parts of the OR expression tree.

This query also contains the following similar predicates shared by other parts of the OR expression tree.

```
    NOT (
        ! Part 1
         (A.LOCATION_1_NAME is null and
          A.LOCATION_2_NAME is null and
          A.LOCATION_3_NAME is null)
        OR
        ! Part 2
         ((A.LOCATION_2_NAME is null and        ! <== shared in PART 1
           A.LOCATION_3_NAME is null)           ! <== shared in PART 1
          and
          ...etc...)
         )              ! end of or
        OR
        ! Part 3
        ((A.LOCATION_3_NAME is null) and        ! <== shared in PART 1 and 2
         ...etc...)
        OR
        ! Part 4
         (...etc...)
        );
```

There is no known workaround for this problem.

This problem has been corrected in Oracle Rdb Release 7.0.7.2.

4.1.12 Zigzag Match Query with Descending Index Segment Returns Wrong Results

Bug 3514413

The following zigzag match query with a descending index segment should select 6 rows but instead just one row is selected.

```
set flags 'strategy,detail';
SELECT   MBR_ID, ACCNT_TYPE, ACCNT_NO, CNTR_PRCE, TRB.SERI_CODE
    FROM   TRD_BOOK_TB TRB, SERIES_DAILY_TB SED, SERIES_TB SER
    WHERE   TRB.DRV_MKT_ID = 'BF'
      AND   TRB.ULY_ID = '61'
      AND   SED.YYMMDD    =   '20040309'
      AND   TRB.SERI_CODE =   SED.SERI_CODE
      AND   SED.SERI_CODE =   SER.SERI_CODE
      AND   SER.ACT_DATE  <=  '20040309'
      AND   SER.DEL_DATE  >=  '20040309'
;
Tables:
```

```
  0 = TRD_BOOK_TB
  1 = SERIES_DAILY_TB
  2 = SERIES_TB
Conjunct: 1.SERI_CODE = 2.SERI_CODE
Match
  Outer loop
    Conjunct: 0.SERI_CODE = 1.SERI_CODE
    Match
      Outer loop
        Sort: 0.SERI_CODE(a)
        Conjunct: (0.DRV_MKT_ID = 'BF') AND (0.ULY_ID = '61')
        Get     Retrieval sequentially of relation 0:TRD_BOOK_TB
      Inner loop      (zig-zag)
        Index only retrieval of relation 1:SERIES_DAILY_TB
          Index name  SERIES_DAILY_IDX [1:1]
            Keys: 1.YYMMDD = '20040309'
  Inner loop
    Temporary relation
    Sort: 2.SERI_CODE(a)
    Conjunct: 2.ACT_DATE <= '20040309'
    Leaf#01 BgrOnly 2:SERIES_TB Card=1002
      Bool: 2.DEL_DATE >= '20040309'
      BgrNdx1 SERIES_IDX [0:1] Fan=11
        Keys: 2.ACT_DATE <= '20040309'
        Bool: 2.DEL_DATE >= '20040309'
 TRB.MBR_ID   TRB.ACCNT_TYPE   TRB.ACCNT_NO    TRB.CNTR_PRCE   TRB.SERI_CODE
 016          3                001000027              106.70   KR4161460000
1 row selected
```

As a workaround, the query works if the zigzag match strategy is disabled, as in the following example.

```
SQL> set flags 'nozigzag_match';
  OR
$DEFINE RDMS$$DISABLE_ZIGZAG_MATCH 2

Here is the new strategy of the new run:

Tables:
  0 = TRD_BOOK_TB
  1 = SERIES_DAILY_TB
  2 = SERIES_TB
Conjunct: 1.SERI_CODE = 2.SERI_CODE
Match
  Outer loop
    Conjunct: 0.SERI_CODE = 1.SERI_CODE
    Match
      Outer loop
        Sort: 0.SERI_CODE(a)
        Conjunct: (0.DRV_MKT_ID = 'BF') AND (0.ULY_ID = '61')
        Get     Retrieval sequentially of relation 0:TRD_BOOK_TB
      Inner loop
        Index only retrieval of relation 1:SERIES_DAILY_TB
          Index name  SERIES_DAILY_IDX [1:1]
            Keys: 1.YYMMDD = '20040309'
  Inner loop
    Temporary relation
    Sort: 2.SERI_CODE(a)
    Conjunct: 2.ACT_DATE <= '20040309'
    Leaf#01 BgrOnly 2:SERIES_TB Card=1002
      Bool: 2.DEL_DATE >= '20040309'
```

```
      BgrNdx1 SERIES_IDX [0:1] Fan=11
        Keys: 2.ACT_DATE <= '20040309'
        Bool: 2.DEL_DATE >= '20040309'
 TRB.MBR_ID   TRB.ACCNT_TYPE   TRB.ACCNT_NO    TRB.CNTR_PRCE   TRB.SERI_CODE
 029          3                001000025              103.90   KR4161430000
 017          0                001000004              103.90   KR4161430000
 029          1                001000023              106.45   KR4161460000
 016          3                001000027              106.45   KR4161460000
 029          2                001000099              106.70   KR4161460000
 016          3                001000027              106.70   KR4161460000
6 rows selected
```

The query also works with the zigag match strategy if the index column YYMMDD of SERIES_DAILY_IDX index is re−defined as ASCENDING instead of DESCENDING.

```
drop index SERIES_DAILY_IDX;
create unique index SERIES_DAILY_IDX
    on SERIES_DAILY_TB (
!   YYMMDD  desc,
    YYMMDD  asc,
    SERI_CODE asc);

Tables:
  0 = TRD_BOOK_TB
  1 = SERIES_DAILY_TB
  2 = SERIES_TB
Conjunct: 1.SERI_CODE = 2.SERI_CODE
Match
  Outer loop
    Conjunct: 0.SERI_CODE = 1.SERI_CODE
    Match
      Outer loop
        Sort: 0.SERI_CODE(a)
        Conjunct: (0.DRV_MKT_ID = 'BF') AND (0.ULY_ID = '61')
        Get     Retrieval sequentially of relation 0:TRD_BOOK_TB
      Inner loop      (zig−zag)
        Index only retrieval of relation 1:SERIES_DAILY_TB
          Index name  SERIES_DAILY_IDX [1:1]
            Keys: 1.YYMMDD = '20040309'
  Inner loop
    Temporary relation
    Sort: 2.SERI_CODE(a)
    Conjunct: 2.ACT_DATE <= '20040309'
    Leaf#01 BgrOnly 2:SERIES_TB Card=1002
      Bool: 2.DEL_DATE >= '20040309'
      BgrNdx1 SERIES_IDX [0:1] Fan=11
        Keys: 2.ACT_DATE <= '20040309'
        Bool: 2.DEL_DATE >= '20040309'
 TRB.MBR_ID   TRB.ACCNT_TYPE   TRB.ACCNT_NO    TRB.CNTR_PRCE   TRB.SERI_CODE
 029          3                001000025              103.90   KR4161430000
 017          0                001000004              103.90   KR4161430000
 029          1                001000023              106.45   KR4161460000
 016          3                001000027              106.45   KR4161460000
 029          2                001000099              106.70   KR4161460000
 016          3                001000027              106.70   KR4161460000
6 rows selected
```

This problem has been corrected in Oracle Rdb Release 7.0.7.2.

4.2 SQL Errors Fixed

4.2.1 COMMIT or ROLLBACK Bugchecks in Conditional Expression

Bug 2350880

In prior versions of Oracle Rdb, attempts to COMMIT or ROLLBACK a transaction within the scope of a conditional expression (IF or WHILE) would cause a bugcheck.

- Alpha OpenVMS 7.3−1
- Oracle Rdb Server V7.0−70
- COSI−F−BUGCHECK, internal consistency failure
- Exception occurred at KOD$PREPARE + 00000228
- Called from KOD$COMMIT + 0000018C
- Called from RDMS$$INT_COMMIT_TRANSACTION + 000002BC

The following shows one example query and the resulting bugcheck dump.

```
SQL> attach 'filename sql$database';
SQL>
SQL> set flags 'trace';
SQL>
SQL> begin
cont> set transaction read only;
cont>
cont> if exists
cont>      (select *
cont>       from rdb$relation_fields t1, rdb$relations t2
cont>       where t2.rdb$relation_name = t1.rdb$relation_name)
cont> then
cont>     trace 'rows exist';
cont>     commit;
cont>     set transaction read only;
cont> end if;
cont>
cont> commit;
cont> end;
~Xt: rows exist
%RDMS−I−BUGCHKDMP, generating bugcheck dump file DISK1:[TEST]RDSBUGCHK.DMP;
```

This problem has been corrected in Oracle Rdb Release 7.0.7.2. Rdb now closes the index scan from the IF and WHILE condition prior to executing the body of the conditional expression.

4.2.2 Unexpected Failure When Using ALTER ... THRESHOLDS Clause

Bug 3283407

In prior releases of Oracle Rdb, the THRESHOLDS clause for ALTER INDEX or ALTER STORAGE MAP was rejected if the logical area already existed.

The following example shows the reported error.

```
SQL> alter index u1_partitioned_table store using (attr1, attr2)
cont> in sa1 ( thresholds are (90,90,90) )  with limit of  (10,15)
cont> in sa2 ( thresholds are (90,90,90) )  with limit of  (10,25)
cont> in sa3 ( thresholds are (90,90,90) )  with limit of  (10,35);
SQL> commit;
SQL> alter index u1_partitioned_table store using (attr1, attr2)
cont> in sa1 ( thresholds are (90,90,90) )  with limit of  (10,15)
cont> in sa2 ( thresholds are (90,90,90) )  with limit of  (10,25)
cont> in sa3 ( thresholds are (90,90,90) )  with limit of  (10,35);
%RDB-E-NO_META_UPDATE, metadata update failed
-RDMS-E-THRESHAREEXI, illegal thresholds usage - area SA1 exists,
and cannot have THRESHOLDS respecified
```

The problem was that the new threshold values were the same as those in that index or storage map partition and could have been ignored by Rdb.

This problem has been corrected in Oracle Rdb Release 7.0.7.2. Rdb now ignores the THRESHOLDS clause if it respecifies the existing thresholds for the index or storage map partition.

4.2.3 Unexpected Bugcheck when Oracle−style Outer Join Used with Subselect

Bug 3329186

When the Oracle style outer join modifier (+) was used in a WHERE clause that also contained a subselect with an aggregate function (SUM, AVG, MIN, MAX, or COUNT), a bugcheck would be generated by SQL.

The following shows a simple example of the type of query.

```
SQL> select *
cont> from employees e, salary_history sh
cont> where e.employee_id (+) = sh.employee_id
cont>   and sh.salary_amount = (select max (salary_amount)
cont>                           from salary_history);
%RDMS-I-BUGCHKDMP, generating bugcheck dump file DISK1:[DATABASE]SQLBUGCHK.DMP;
%SQL-F-BUGCHK, There has been a fatal error. Please contact your Oracle
support representative. SQL$SEMRSE - 71
```

The only workaround is to recode the query using ANSI/ISO Standard SQL syntax using the LEFT OUTER JOIN or RIGHT OUTER JOIN operators.

This problem has been corrected in Oracle Rdb Release 7.0.7.2.

4.2.4 %SQL−F−NODBFILE, Alias Missing a Declaration With SQLMOD

Bug 1258536

This problem can arise when a SQL$MOD module is compiled with /CONNECT and has a database alias declared with a run−time resolution. In this case, if another SQL$MOD module or a Dynamic SQL module

declares the same alias but with a compile−time resolution, the first call to any procedure in the module with the run−time resolution will fail with:

```
SQL−F−NODBFILE, ALIAS <alias−name> IS MISSING A DECLARATION
where <alias−name> will be the alias that fails.
```

For example, suppose SQL$MOD module1 declares an alias with run−time resolution like this:

```
declare base alias for compiletime filename 'mf_personnel'
     runtime filename db_name
Note: procedures in module1 will have "db_name" as a parameter
```

Suppose also that SQL$MOD module2 declares the same alias but with a compile−time resolution like this:

```
declare alias base filename 'mf_personnel'
```

Now both modules are linked with a main program that calls a procedure from module1. The call to the module1 procedure will result in:

```
SQL−F−NODBFILE, ALIAS BASE IS MISSING A DECLARATION
```

There is no known workaround for this problem. There is no way to achieve run−time resolution of the alias.

This problem has been corrected in Oracle Rdb Release 7.0.7.2.

4.3 Oracle RMU Errors Fixed

4.3.1 RMU/CHECKPOINT Slow When Number of Nodes is One

Bug 3240378

If a database was set to have NUMBER OF CLUSTER NODES 1, and the AIJ log server (ALS) was not enabled, and there was no update activity occurring at that point in time, the RMU/CHECKPOINT command could take many minutes to complete. For example:

```
$ SQL$
  CREATE DATABASE FILENAME CKPT_TEST
    NUMBER OF CLUSTER NODES 1
    CREATE STORAGE AREA RDB$SYSTEM FILENAME CKPT_TEST;
  DISCONNECT ALL;

  ALTER DATABASE FILE CKPT_TEST
    JOURNAL IS ENABLED
      (FAST COMMIT IS ENABLED,
       -- This test only fails if there is no ALS.
       LOG SERVER IS MANUAL)
    ADD JOURNAL AIJ_1 FILENAME 'SYS$DISK:[]CKPT_TEST.AIJ';

  EXIT;
$ CREATE SUB1.COM
$ SQL$
  ATTACH 'FILENAME CKPT_TEST';
  INSERT INTO T1 VALUES (1);
  COMMIT;
  -- Wait till we are killed.
  $WAIT 1:0:0
$
$ SPAWN /NOWAIT/OUT=SUB1.LOG/PROCESS=CKPT_TIMELY_SUB @SUB1.COM
%DCL-S-SPAWNED, process CKPT_TIMELY_SUB spawned
$ WAIT 0:1:0
$ START_TIME = F$CVTIME (,,"MINUTEOFYEAR")
$ RMU/CHECKPOINT CKPT_TEST
$ END_TIME = F$CVTIME (,,"MINUTEOFYEAR")
$ CKPT_DURATION = END_TIME - START_TIME
$ IF CKPT_DURATION .GT. 2
$ THEN WRITE SYS$OUTPUT -
  "''CKPT_DURATION' minutes is too long for an RMU/CHECKPOINT command"
5 minutes is too long for an RMU/CHECKPOINT command
$ ENDIF
$ STOP CKPT_TIMELY_SUB
```

This problem can be avoided by utilizing the AIJ log server process or by setting the database number of cluster nodes greater than one.

This problem has been corrected in Oracle Rdb Release 7.0.7.2.

4.3.2 RMU/LOAD/DEFER_INDEX_UPDATES ACCVIO with a Hashed Partitioned Index

Bug 3262298

If an RMU/LOAD/DEFER_INDEX_UPDATES command was used and a partitioned hashed index was the only or primary index defined for the table being loaded, an access violation would occur and an RMUBUGCHK.DMP and RDSBUGCHK.DMP would be generated. The exception in the RDSBUGCHK.DMP would occur in the routine PSII$INSERT_T. This was because the test for a hashed index did not work correctly so an attempt was made to update a hashed index using a sorted index routine. This problem has been fixed.

The following example shows that an access violation occurred if the /DEFER_INDEX_UPDATES qualifier was used for an RMU/LOAD of a table for which the only or the primary index was a partitioned hashed index.

```
SQL> att 'fi [.to]mf_personnel';
SQL> show table job_history
Information for table JOB_HISTORY

Columns for table JOB_HISTORY:
Column Name                     Data Type        Domain
-----------                     ---------        ------
EMPLOYEE_ID                     CHAR(5)          ID_NUMBER
JOB_CODE                        CHAR(4)          JOB_CODE
JOB_START                       DATE VMS         STANDARD_DATE
JOB_END                         DATE VMS         STANDARD_DATE
DEPARTMENT_CODE                 CHAR(4)          DEPARTMENT_CODE
SUPERVISOR_ID                   CHAR(5)          ID_NUMBER

Table constraints for JOB_HISTORY:
No constraints found

Constraints referencing table JOB_HISTORY:
No constraints found

Indexes on table JOB_HISTORY:
JOB_HISTORY_HASH                with column EMPLOYEE_ID
  Duplicates are allowed
  Type is Hashed Scattered
  Compression is DISABLED
Store clause:           STORE
                             using (EMPLOYEE_ID)
                                 in EMPIDS_LOW
                                     with limit of ('00200')
                                 in EMPIDS_MID
                                     with limit of ('00400')
                                 otherwise in EMPIDS_OVER
 Comment:        hash index for job_history

Storage Map for table JOB_HISTORY:
     JOB_HISTORY_MAP

Triggers on table JOB_HISTORY:
No triggers found

SQL> exit
```

```
$rmu/load/defer/log [.to]mf_personnel job_history job_history.unl
%RDMS-I-BUGCHKDMP, generating bugcheck dump file
 DEVICE:[LOGIN]RDSBUGCHK.DMP;
%SYSTEM-F-ACCVIO, access violation, reason mask=00,
 virtual address=00000050, PC=00186C66, PSL=03C00000
%RMU-I-BUGCHKDMP, generating bugcheck dump file
 DEVICE:[LOGIN]RMUBUGCHK.DMP;
%RMU-I-DATRECREAD,  274 data records read from input file.
%RMU-I-DATRECSTO,   0 data records stored.
%RMU-F-FTL_LOAD, Fatal error for LOAD operation at 13-JAN-2004 13:32:24.76
$type [login]RDSBUGCHK.DMP

***** Exception at 0093EBB0 : PSII$INSERT_T + 000000B7
```

The workaround for this problem is to not use the qualifier /DEFER_INDEX_UPDATES when doing the RMU/LOAD of a table for which the only or primary index is a partitioned hashed index.

This problem has been corrected in Oracle Rdb Release 7.0.7.2.

4.3.3 RMU/RECOVER/ONLINE Without /JUST_CORRUPT or /AREA Qualifiers Bugchecks

As described in the documentation, when issuing an RMU/RECOVER command, the /ONLINE qualifier can only be use in conjunction with the /JUST_PAGE or /AREA qualifier.

In previous releases, if the /ONLINE qualifier was used alone, it resulted in an RMU bugcheck as shown below.

```
$ rmu/recover/log/online AIJ_BCK.AIJ
%RMU-E-RECFAILED, fatal, unexpected roll-forward error detected at AIJ record 1
%COSI-F-BUGCHECK, internal consistency failure
%RMU-F-FATALOSI, Fatal error from the Operating System Interface.
%RMU-I-BUGCHKDMP, generating bugcheck dump file disk:[dir]RMUBUGCHK.DMP;
%RMU-F-FTL_RCV, Fatal error for RECOVER operation at <timestamp>
```

The exception and the call frame in the dump are as follows.

```
***** Exception at 007212B8 : KUTREC$ABORT + 000005D8
%COSI-F-BUGCHECK, internal consistency failure
Saved PC = 0071F5E4 : KUTREC$SETUP + 00000214
Saved PC = 00474198 : RMUREC$RECOVER_ACTION + 00000948
Saved PC = 004737D8 : RMUCLI$RECOVER + 000004C8
Saved PC = 003A56F4 : RMU_DISPATCH + 00000D44
Saved PC = 003A4508 : RMU_STARTUP + 000004A8
Saved PC = 001E002C : RMU$MAIN + 0000002C
```

Now RMU will detect missing /JUST_PAGE or /AREA qualifiers when /ONLINE is used and will return an error message, as in the following example.

```
$ rmu/recover/log/online AIJ_BCK.AIJ
%RMU-F-CONFLSWIT, conflicting qualifiers /ONLINE and /AREAS or /JUST_CORRUPT
missing
%RMU-F-FTL_RCV, Fatal error for RECOVER operation at <timestamp>
```

This problem has been corrected in Oracle Rdb Release 7.0.7.2.

4.3.4 Invalid RMU−E−INASPAREA Message in RMU/VERIFY/ROOT

Bug 3424732

While doing an RMU/VERIFY of the database root, each live storage area entry in the database root was correctly reloaded to get its current state on all cluster nodes accessing the database but the corresponding snapshot area entry was not reloaded by RMU/VERIFY to bring it up to date. If a new database storage area had just been created on one node in the cluster, this could lead to the following incorrect RMU/VERIFY error message being output on another cluster node which was doing an RMU/VERIFY of the database root.

```
RMU−E−INASPAREA, Live area AREA_NAME points to a snapshot area that is
inactive.
```

Repeating the RMU/VERIFY would not show this error since this would reload the snapshot entry and bring it up to date. This problem has been fixed.

The following example shows the sequence of adding a storage area entry to a database on one node and then verifying the database on another node that gave the RMU−E−INASPAREA error.

On one cluster node...

```
$ @RDM$DEMO:PERSONNEL SQL M NOCDD
$ SQL
SQL> ALTER DATABASE FILENAME MF_PERSONNEL OPEN IS MANUAL;
SQL> ALTER DATABASE FILENAME MF_PERSONNEL RESERVE 1 STORAGE AREA;
SQL> EXIT;
$ RMU/OPEN MF_PERSONNEL
```

On another node in the cluster...

```
$ RMU/OPEN MF_PERSONNEL
```

Back to first cluster node...

```
$ SQL
SQL> ALTER DATABASE FILENAME MF_PERSONNEL ADD STORAGE AREA TEST;
SQL> EXIT;
```

Back to the second cluster node...

```
$ RMU/VERIFY MF_PERSONNEL
%RMU−E−INASPAREA, Live area TEST points to a snapshot area that is inactive.
```

The workaround for this problem is do the RMU/VERIFY on the same cluster node where the storage area was created or to repeat the RMU/VERIFY on the second cluster node if the RMU−E−INASPAREA error is returned on the first RMU/VERIFY.

This problem has been corrected in Oracle Rdb Release 7.0.7.2.

4.3.5 RMU Support For /DENSITY = SDLT320

Oracle Rdb RMU commands that support the /DENSITY qualifier (ie, RMU/BACKUP, RMU/BACKUP/AFTER_JOURNAL and RMU/OPTIMIZE_AIJ) now support the keyword "SDLT320" for use with SuperDLT320 tape drives.

4.3.6 Incorrect Verification of Invalid Reference DBKEYS in Ranked Indices

Bugs 3009262 and 3537202

If an index of *TYPE IS SORTED RANKED* had invalid reference dbkeys in duplicate overflow nodes, RMU/VERIFY/INDEX could fail to report the error. If the error was reported, it was reported as an informational message.

This type of corruption was possibly introduced by the problem referenced in Section 4.1.5.

In the following example, the index corruption is correctly reported.

```
$ RMU/VERIFY/INDEX=MY_IDX MY_DB
%RMU−E−BADREFDBK, Invalid reference pointer 358:263034:0 for duplicate B−tree
node 299:88538:0
%RMU−I−DUPRECDBK, the last duplicate record dbkey was 358:294862:1
%RMU−I−DUPOWNDBK, Dbkey of owner of this duplicate node is 299:89267:1
%RMU−I−BTRERPATH, parent B−tree node of 299:89267:1 is at 299:89245:0
%RMU−I−BTRERPATH, parent B−tree node of 299:89245:0 is at 299:89138:1
%RMU−I−BTRERPATH, parent B−tree node of 299:89138:1 is at 299:88639:1
%RMU−I−BTRERPATH, parent B−tree node of 299:88639:1 is at 299:88437:1
%RMU−I−BTRROODBK, root dbkey of B−tree is 299:88437:1
```

This problem represents a real corruption in the index and can lead to bugchecks as described in Section 4.1.5.

As a workaround for this problem, the offending index must be dropped and recreated.

This problem has been corrected in Oracle Rdb Release 7.0.7.2.

4.4 LogMiner Errors Fixed

4.4.1 RMU /UNLOAD /AFTER_JOURNAL Incorrect NULL Bit Setting When VARCHAR is Last Column

Bug 3309002

In prior versions of Oracle Rdb, the RMU /UNLOAD /AFTER_JOURNAL command could incorrectly process the null bit vector for tables with the last column being a VARCHAR data type. The LogMiner was not correctly calculating the position of the null bit vector within the data record and could pick up stray bit patterns as the null bit vector content. The effect of not using the correct null bit vector content could be NULL column values being incorrectly returned as not NULL or not NULL column values being incorrectly returned as NULL.

The following example script demonstrates one possible effect of this problem. The column Z$I6 should be returned as NULL but is being extracted as a zero value:

```
$ SQL$
  CREATE DATA FILE FOO;
  CREATE TABLE T1 (
    Z$I1 INT, Z$I2 INT, Z$I3 INT, Z$I4 INT,
    Z$I5 INT, Z$I6 INT, X$I7 INT, Z$I8 INT,
    Z$I9 INT, Z$I10 INT, Z$V1 VARCHAR(10));
  COMMIT;
  DISCONNECT ALL;
  ALTER DATA FILE FOO ADD JOURNAL J1 FILE J1 JOURNAL ENABLE;
  EXIT;
$ RMU/SET LOGMINER/ENABLE FOO.RDB
$ RMU/BACKUP/NOLOG FOO.RDB NLA0:FOO
$ RMU/BACKUP/AFTER/NOLOG FOO.RDB NLA0:FOO
$ SQL$
  ATTACH 'FILE FOO';
  INSERT INTO T1 VALUES (1,2,3,4,5,NULL,7,8,9,10,'TEST');
1 ROW INSERTED
  COMMIT;
  EXIT;
$ RMU/BACKUP/AFTER/NOLOG FOO.RDB B1.AIJ
$ RMU/UNLOAD/AFTER_JOURNAL/NOLOG FOO.RDB B1.AIJ -
  /TABLE=(NAME=T1,OUTPUT=T1.TXT) /FORMAT=DUMP
$ SEARCH T1.TXT Z$
Z$I1                            : 1
Z$I2                            : 2
Z$I3                            : 3
Z$I4                            : 4
Z$I5                            : 5
Z$I6                            : 0
Z$I8                            : 8
Z$I9                            : 9
Z$I10                           : 10
Z$V1                            : (4) TEST
$ EXIT
```

This problem has been corrected in Oracle Rdb Release 7.0.7.2. The RMU /UNLOAD /AFTER_JOURNAL command now correctly determines the location of the null bit vector within the record when the final column

of the record is a VARCHAR.

4.5 Oracle Trace Errors Fixed

4.5.1 Oracle TRACE COLLECT FORMAT Command Could Not Create a Database After Rdb Upgrade

There was a problem with Oracle Trace Release 2.4.1 where, after upgrading to Rdb Release 7.0.7.1, the COLLECT FORMAT command was no longer able to successfully create the database where the formatted data is stored. This problem has been fixed in Oracle TRACE Release 2.4.1.1 Update 01. Note that this problem has been fixed in Oracle TRACE, not Oracle Rdb.

The following example shows the problem that the Oracle TRACE COLLECT FORMAT command had creating a database when Rdb was upgraded to Release 7.0.7.1.

```
$ COLLECT FORMAT SAMPLE_DATA.DAT SAMPLE_DATA.RDB
%EPC-E-FMT_FAILURE, Formatting failed
-RDMS-F-NOEUACCESS, unable to acquire exclusive access to database
%EPC-E-OPFAIL, Operation failed
```

This problem has been corrected in Oracle TRACE Release 2.4.1.1 Update 01.

Chapter 5
Software Errors Fixed in Oracle Rdb Release 7.0.7.1

This chapter describes software errors that are fixed by Oracle Rdb Release 7.0.7.1.

5.1 Software Errors Fixed That Apply to All Interfaces

5.1.1 Recovery of Empty Optimized AIJ Does Not Update the Sequence Number

Bug 2581948

The recovery of an empty optimized AIJ does not update the next AIJ sequence number. This will prevent the recovery of the next AIJ as shown below. An optimized AIJ file can be empty if the corresponding AIJ file contains only rolled−back transactions. See the following example.

```
$ sql$
SQL> attach 'filename opt_aij';
SQL> insert into t1 values (1);
1 row inserted
SQL> rollback;
SQL> exit
$ RMU /BACKUP /AFTER OPT_AIJ OPT_AIJ_BCK1

− Note that opt_aij_bck1 contains a single rolled−back transaction

$ sql$
SQL> attach 'filename opt_aij';
SQL> insert into t1 values (2);
1 row inserted
SQL> commit;
SQL> exit
$ RMU /BACKUP /AFTER OPT_AIJ OPT_AIJ_BCK2

! opt_aij_bck2 contains a single committed transaction

$ RMU /OPTIMIZE /AFTER OPT_AIJ_BCK1 OPT_AIJ_OPT1
$ RMU /OPTIMIZE /AFTER OPT_AIJ_BCK2 OPT_AIJ_OPT2
$ RMU /RESTORE /NOCDD OPT_AIJ

! Recovering the empty Optimized AIJ file opt_aij_opt1

$ RMU /RECOVER /LOG OPT_AIJ_OPT1.OAIJ
%RMU−I−LOGRECDB, recovering database file $111$DUA4:[VIGIER.OPTAIJ.DB]OPT_AIJ.RDB;1
%RMU−I−LOGOPNAIJ, opened journal file RAID1:[VIGIER.OPTAIJ.DB]OPT_AIJ_OPT1.OAIJ;1 at  8−JAN−200
%RMU−I−AIJONEDONE, AIJ file sequence 0 roll−forward operations completed
%RMU−I−LOGRECOVR, 0 transactions committed
%RMU−I−LOGRECOVR, 0 transactions rolled back
%RMU−I−LOGRECOVR, 0 transactions ignored
%RMU−I−AIJNOACTIVE, there are no active transactions
%RMU−I−AIJSUCCES, database recovery completed successfully
%RMU−I−AIJNXTSEQ, to continue this AIJ file recovery, the sequence number needed will be 0
%RMU−I−AIJALLDONE, after−image journal roll−forward operations completed
%RMU−W−NOTRANAPP, no transactions in this journal were applied
%RMU−I−AIJSUCCES, database recovery completed successfully
%RMU−I−AIJFNLSEQ, to start another AIJ file recovery, the sequence number needed will be 0
%RMU−I−AIJNOENABLED, after−image journaling has not yet been enabled

− Note that the next AIJ Sequence number has been left as 0

− Recovering the second Optimized AIJ file opt_aij_opt2
```

```
$ RMU /RECOVER /LOG OPT_AIJ_OPT2.OAIJ
%RMU-I-LOGRECDB, recovering database file $111$DUA4:[VIGIER.OPTAIJ.DB]OPT_AIJ.RDB;1
%RMU-F-AIJNORCVR, recovery of this journal must start with sequence 0
%RMU-F-FTL_RCV, Fatal error for RECOVER operation at  8-JAN-2003 10:23:38.32
```

The recovery fails since it does not have the expected AIJ sequence number.

A workaround is to put all the optimized AIJ files in the same command line.

```
$ RMU /RECOVER /LOG OPT_AIJ_OPT1.OAIJ,OPT_AIJ_OPT2.OAIJ
```

This problem has been corrected in Oracle Rdb Release 7.0.7.1.

5.1.2 Left Outer Join Query With OR Predicate Returns Wrong Results

Bugs 2836144 and 1837522

The following left outer join query with an OR predicate, having an equality predicate of constant values on the left side, and another equality predicate of a column and a constant value on the right side, returns the wrong result. It should find 274 rows but it finds only one row.

```
set flags 'strategy,detail';
select count(*) from
  job_history as c1
  left outer join
  employees as c2 on (c1.employee_id = c2.employee_id)
  where
      1=1 OR c1.job_code = 'JNTR';
Tables:
  0 = JOB_HISTORY
  1 = EMPLOYEES
Aggregate: 0:COUNT (*)
Conjunct: 0.JOB_CODE = 'JNTR'
Conjunct: 0.JOB_CODE = 'JNTR'
Match    (Left Outer Join)
  Outer loop
    Conjunct: (1 = 1) OR (0.JOB_CODE = 'JNTR')
    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]

           1
1 row selected
```

This problem was partially fixed in Bug 1837522 where the equality contains one column and one constant.

As a workaround, the query works if the left and right side of the OR predicate is swapped as in the following example.

```
select count(*) from
  job_history as c1
  left outer join
```

```
  employees as c2 on (c1.employee_id = c2.employee_id)
  where
      c1.job_code = 'JNTR' OR 1=1;
Tables:
  0 = JOB_HISTORY
  1 = EMPLOYEES
Aggregate: 0:COUNT (*)
Conjunct: 0.JOB_CODE = 'JNTR'
Conjunct: 0.JOB_CODE = 'JNTR'
Match    (Left Outer Join)
  Outer loop
    Conjunct: (0.JOB_CODE = 'JNTR') OR (1 = 1)
    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]

         274
1 row selected
```

This problem has been corrected in Oracle Rdb Release 7.0.7.1.

5.1.3 Left Outer Join Query With OR Predicate Returns Wrong Results

Bugs 2845172, 2836144 and 1837522

The following left outer join query with an OR predicate, having an equality predicate of constant values on the left side, and another equality predicate of a column and a constant value on the right side, returns the wrong result. It should find 274 rows, but it finds only 1 row.

```
set flags 'strategy,detail';
select
    t1.home_area_cd,t1.home_ph_num,
    t1.bus_area_cd,t1.bus_ph_num
  from
    t1 left outer join t2  on t1.acct_num = t2.acct_num
  where
    (t1.home_area_cd=999) OR
    (t1.bus_area_cd=310 and t1.bus_ph_num=5355400) ;
Tables:
  0 = T1
  1 = T2
Conjunct: 0.HOME_AREA_CD = 999              <== Note1 : MISSING OR predicate
Conjunct: (0.BUS_AREA_CD = 310) AND (0.BUS_PH_NUM = 5355400)
Cross block of 2 entries        (Left Outer Join)
  Cross block entry 1
    OR index retrieval
      Conjunct: 0.HOME_AREA_CD = 999
      Get     Retrieval by index of relation 0:T1
        Index name  T1_IDX1 [1:1]
          Keys: 0.HOME_AREA_CD = 999
      Conjunct: NOT (0.HOME_AREA_CD = 999) AND (0.BUS_AREA_CD = 310) AND (
                0.BUS_PH_NUM = 5355400)
      Get     Retrieval by index of relation 0:T1
        Index name  T1_IDX2 [2:2]
          Keys: (0.BUS_AREA_CD = 310) AND (0.BUS_PH_NUM = 5355400)
```

```
  Cross block entry 2
    Conjunct: 0.ACCT_NUM = 1.ACCT_NUM
    Get     Retrieval sequentially of relation 1:T2
0 rows selected
```

Note1: Notice that the OR predicate is missing between the two conjuncts.

This problem is similar to the query in Bug 2836144, but the only difference is that this query uses static OR retrieval strategy.

As a workaround, the query works if the left and right side of the OR predicate is swapped in this example but it won't do the trick with the customer's original query where the OR predicate has an additional equality predicate. For example:

```
select
    t1.home_area_cd,t1.home_ph_num,
    t1.bus_area_cd,t1.bus_ph_num
  from
    t1 left outer join t2  on t1.acct_num = t2.acct_num
  where
    (t1.home_area_cd=999 and t1.home_ph_num=999) or
    (t1.bus_area_cd=310 and t1.bus_ph_num=5355400);
```

This problem has been corrected in Oracle Rdb Release 7.0.7.1.

5.1.4 Query Bugchecks When IN Clause Contains More Than Two DBKEYS

Bug 2742592

The following query bugchecks when the IN clause contains more than 2 dbkeys.

```
create table prova (i integer, c char(32));
insert into prova values (1,'one');
insert into prova values (2,'two');
insert into prova values  (3,'three');

select * from prova where dbkey in (:dbk1,:dbk2,:dbk3);
declare :dbk1 char(8);
declare :dbk2 char(8);
declare :dbk3 char(8);
declare c1 table cursor for select dbkey from prova;
open c1;
fetch c1 into :dbk1;
fetch c1 into :dbk2;
fetch c1 into :dbk3;
close c1;
```

The following query works.

```
select * from prova where dbkey in (:dbk1,:dbk2);
OR index retrieval
  Conjunct        Firstn  Get     Retrieval by DBK of relation PROVA
  Conjunct        Firstn  Get     Retrieval by DBK of relation PROVA
           I   C
```

```
           1   one
           2   two
2 rows selected
```

However, with more than 2 dbkeys in the IN clause, it bugchecks.

```
select * from prova where dbkey in (:dbk1,:dbk2,:dbk3);
%RDMS-I-BUGCHKDMP, generating bugcheck dump file USD04:[ONG]RDSBUGCHK.DMP;
```

There is no known workaround for this problem.

This problem has been corrected in Oracle Rdb Release 7.0.7.1.

5.1.5 Processes Loop at IPL2 When VLM Feature Used

Bug 2859466

It was possible for processes to become stuck in a loop in the Oracle Rdb image RDMPRV.EXE at VMS interrupt priority level (IPL) 2. Because the process was looping at an elevated IPL, it was not possible to delete the process. A system reboot was necessary to get rid of the looping processes. This problem only occurred when the Very Large Memory (VLM) feature was utilized for database shared memory or row caches.

Often the initial symptom of this problem was unresolved database stalls. Because the looping processes often held database locks, and since the looping prevented the processes from responding to blocking AST requests, other database processes would stall waiting for the looping processes to release their locks. Analysis of the processes not responding to blocking AST requests revealed that those processes were looping in the RDMPRV.EXE image.

To avoid this problem, disable the VLM feature for database shared memory or row caches.

This problem has been corrected in Oracle Rdb Release 7.0.7.1.

5.1.6 DBR Bugchecks at DBR$DDTM_RESOLVE + 000003F4

The database recovery process (DBR) would sometimes bugcheck with the following failure:

```
***** Exception at 0005EC94 : DBR$DDTM_RESOLVE + 000003F4
%COSI-F-BUGCHECK, internal consistency failure
```

This bugcheck would occur when the OpenVMS system service $GETDTIW would return an undocumented transaction state value (11, or "IN_DOUBT") for an unresolved transaction. The DBR would fail since it was not expecting that transaction state.

Subsequent attempts to open or attach to the database would usually succeed. The second query using the $GETDTIW would typically return an expected state value.

This problem has been corrected in Oracle Rdb Release 7.0.7.1.

5.1.7 Replication Option and LogMiner Features Active at the Same Time

Bug 2903092

When both the Replication Option and LogMiner(TM) features are enabled, it is possible for the "pre−delete" record contents stored in the after−image journal file for the LogMiner to contain incorrect contents. This problem may be indicated by errors from the RMU /UNLOAD /AFTER_JOURNAL command such as the following example:

```
%RMU−W−RECVERDIF, Record at DBKEY 819:20615:8 in table
 "FOOBAR" version 262 does not match current version.
```

This problem was caused by an incorrect buffer being used when writing the "pre−delete" record contents for the LogMiner. This buffer was also used by the Replication Option code path and the existing saved content was lost.

This problem has been corrected in Oracle Rdb Release 7.0.7.1. A different buffer is now used to save "pre−delete" record content data.

5.1.8 RMU /UNLOAD /AFTER_JOURNAL Created .RRD Content Clarification

Bug 2916639

An optional "RECORD_DEFINITION" keyword can be used with the RMU /UNLOAD /AFTER_JOURNAL command to create a template .RRD (record definition) file that can be used to load a transaction table. The TABLE_DEFINITION keyword can also be used to create a template SQL procedure to create such a transaction table.

The behaviour of the RMU /UNLOAD /AFTER_JOURNAL command when used with these keywords is to append the string "RDB_LM_" to the table name to create the record name in the .RRD or .SQL file.

However, when the existing table name exceeds 24 characters in length, the resultant name for the transaction table in the .SQL or .RRD file exceeds 31 characters and is no longer a valid table name in an Rdb database. In these cases, a decision about the table name to be used must be made and the .RRD or .SQL file must be manually modified.

Oracle Rdb engineering is considering alternatives for future releases to help reduce the impact of this behaviour.

5.1.9 Page Locks Not Released When LOCKING IS PAGE LEVEL

Bug 2959599

When the LOCKING IS PAGE LEVEL storage area attribute was enabled, it was possible for a process to obtain a page lock and not release it when a blocking AST was delivered by another process. Other processes would stall waiting for the page until the owning process removed the page from its buffer pool. The process could continue to hold the lock even after committing its transaction.

When this problem occurred, the output from the ***RMU /SHOW LOCKS /MODE=CULPRIT*** would be similar to the following:

```
================================================================================
SHOW LOCKS/LOCK/MODE=CULPRIT Information
================================================================================

--------------------------------------------------------------------------------
Resource: page 4443

         ProcessID Process Name    Lock ID   System ID Requested Granted
         --------- --------------- --------- --------- --------- -------
Blocker: 2541851A  DKOM_TSG_004... 5300BA60  0001002A            PW
Waiting: 25418513  DKOM_BRZ_PAS... 48001A32  0001002A  PW        NL
```

The OpenVMS system analyzer (SDA) utility SHOW LOCK command would show output similar to the following:

```
Lock id:  5300BA60            PID:    0061011A   Flags: VALBLK  CONVERT NOQUEUE
Par. id:  21010D65            SUBLCKs:       0          SYNCSTS SYSTEM  PROTECT
LKB:      FFFFFFFF.7E352E50   BLKAST: 00000000
Priority:     0000

Granted at    PW   00000000-FFFFFFFF

Resource:          0000115B 00000050  P...[...  Status: PROTCT
 Length   08       00000000 00000000  ........
 Exec. mode        00000000 00000000  ........
 System            00000000 00000000  ........

Local copy
```

Note that in the above output, the blocking AST address (BLKAST) is zero and the lock is granted in PW mode.

To avoid this problem, set the storage area to LOCKING IS ROW LEVEL.

This problem has been corrected in Oracle Rdb Release 7.0.7.1.

5.1.10 Looping or Bugchecks in DIO$FETCH_DBKEY for SORTED RANKED Indexes

Bugs 2936413 and 3017913

If a query specified a range of key values that fell between any existing key values in a SORTED RANKED index and the dynamic optimizer was used, it was possible to experience looping or bugchecks.

If a query specified a range of key values such that the range appeared in upper level index nodes but not in lower level index nodes, it was possible that Oracle Rdb could enter an infinite loop in the routines PSII2FINDSEPINTERVAL2 and PSII2EXPANDNODESEPS2.

If a query specified an empty range on an index, it was possible that bugchecks could occur such as:

```
SYSTEM-F-ACCVIO, access violation
```

```
Exception occurred at DIO$FETCH_DBKEY + 00000200
Called from PSII2FINDSEPINTERVAL2 + 0000015C
Called from PSII2ESTIMATECARD2 + 0000008C
Called from RDMS$$NDX_ESTIM + 000003B0
```

An empty range is one such as that specified by the condition *WHERE MY_FIELD > 'A' AND MY_FIELD < 'B'*. If MY_FIELD were a CHAR(1) field, it would be impossible to find a key value to match the specified condition.

In some cases, Rdb would fail to detect that the range was empty and would try and persue index estimation for the dynamic optimizer. This could result in Rdb attempting to read an erroneous dbkey, or to repeatedly re−read the same dbkey as the next index node.

Oracle Rdb now correctly detects an empty range and returns an estimate of zero rows for the index estimation.

The bugcheck problem can be avoided by specifying ranges that include at least one possible key value. This key value need not exist in the database.

The looping problem can be avoided by rebuilding the offending index or by disabling the dynamic optimizer.

This problem only occurs on indexes of *TYPE IS SORTED RANKED*. So another workaround is to use indexes of *TYPE IS HASHED* or *TYPE IS SORTED* if appropriate.

This problem has been corrected in Oracle Rdb Release 7.0.7.1.

5.1.11 RMU/CLOSE/WAIT Hangs Waiting for ALS to Terminate

It was possible for the ***RMU/CLOSE/WAIT*** command to hang if the database had LOG SERVER IS AUTOMATIC specified and a user attempted to attach to the database immediately after the ***RMU/CLOSE*** command was issued. This problem was introduced in Oracle Rdb Release 7.0.7.

The ***RMU/CLOSE/WAIT*** command will ensure that database recovery processes (DBRs) are executed before closing the database. In Release 7.0.7, if a user attempted to attach to the database while the DBR process was executing, when the DBR process completed, the database monitor would incorrectly start a new AIJ Log Server (ALS) process. The monitor would then wait for the ALS process to terminate before responding to the ***RMU/CLOSE/WAIT*** command. Since the ALS process had never been instructed to terminate, the database shutdown would never complete and no response was ever delivered to the RMU process that issued the close request.

The only way to avoid this problem is to prevent users from attempting to attach to the database immediately after it has been closed. When the problem occurs, the ALS process must be manually deleted by using the DCL ***STOP/IDENTIFICATION*** command or by issuing another ***RMU/CLOSE*** command with the ***/ABORT=DELPRC*** qualifier.

This problem has been corrected in Oracle Rdb Release 7.0.7.1.

5.1.12 Query With EXISTS Clause and COMPUTED BY Column Returns Wrong Results

Bug 2452636

The following query, checking for EXISTS clause with an equality predicate involving a column of "COMPUTED BY", returns wrong results when applying a match strategy.

```
Columns for table T2:
Column Name         Data Type    Domain
-----------         ---------    ------
A_DATE                          DATE VMS
A_SEC_NAME                      CHAR(12)
A_DET_METHOD                    INTEGER
A_CALC_PRICE                    INTEGER
A_BAD_CMPBY                     INTEGER
 Computed:   by
      case
           when (A_CALC_PRICE > 0 and
                A_CALC_PRICE =
                        (select T0.A_PRICE from T0
                             where T0.A_SEC_NAME = T0.A_SEC_NAME
                                and T0.A_DATE = T0.A_DATE
                                and T0.A_PRICE_SOURCE = 'GIC'
                                and T0.A_PRICE_TYPE = 'STL' limit to 1 rows))
                               then A_DET_METHOD
                                   else 5
                                      end

Indexes on table T2:
T2_NDX                          with column A_SEC_NAME
                and column A_DATE

set flags 'strategy,detail';
sel a_data from T1 s
where exists (select * from T2 p
                where p.A_SEC_NAME=s.A_SEC_NAME and
                      p.a_date=s.a_date and
                      p.a_bad_cmpby =   2);
Tables:
  0 = T1
  1 = T2
  2 = T0
Conjunct: <agg0> <> 0
Match
  Outer loop
    Sort: 0.A_SEC_NAME(a), 0.A_DATE(a)
    Cross block of 2 entries
      Cross block entry 1
        Aggregate: 1:VIA (2.A_PRICE)
        Firstn: 1
        Leaf#01 FFirst 2:T0 Card=10
          Bool: (2.A_SEC_NAME = 2.A_SEC_NAME) AND (2.A_DATE = 2.A_DATE) AND (
                2.A_PRICE_SOURCE = 'GIC') AND (2.A_PRICE_TYPE = 'STL')
          BgrNdx1 T0_NDX [2:2] Fan=9
            Keys: (2.A_PRICE_SOURCE = 'GIC') AND (2.A_PRICE_TYPE = 'STL')
            Bool: (2.A_SEC_NAME = 2.A_SEC_NAME) AND (2.A_DATE = 2.A_DATE)
      Cross block entry 2
        Leaf#02 BgrOnly 0:T1 Card=2
          BgrNdx1 T1_NDX [0:0] Fan=10
  Inner loop      (zig-zag)
    Aggregate-F1: 0:COUNT-ANY (<subselect>)
    Get     Retrieval by index of relation 1:T2
```

```
      Index name  T2_NDX [0:0]
      A_DATA
          14
          10
2 rows selected
```

Notice that the following equality predicate, "p.a_bad_cmpby = 2", involving the "Computed by" column A_BAD_CMPBY is missing in the above strategy and thus, the query returns the wrong result of 2 rows. P.a_bad_cmpby represents the following CASE statement:

```
case
     when (A_CALC_PRICE > 0 and
          A_CALC_PRICE =
                 (select T0.A_PRICE from T0
                     where T0.A_SEC_NAME = T0.A_SEC_NAME
                        and T0.A_DATE = T0.A_DATE
                        and T0.A_PRICE_SOURCE = 'GIC'
                        and T0.A_PRICE_TYPE = 'STL' limit to 1 rows))
      then A_DET_METHOD
       else 5
end
```

Notice that the select statement references only a single table T0.

The missing conjunct is supposed to be generated in the following format with the aggregate value represented by <agg0>.

```
Conjunct: CASE (WHEN ((1.A_CALC_PRICE > 0) AND (1.A_CALC_PRICE = <agg0>))
                THEN 1.A_DET_METHOD ELSE 5) = 2
```

The reason that the conjunct is not created in the query is that the inner match leg contains only the context "1:T2" while the conjunct involves an aggregate value with external context "0:T0" from the outer match leg.

Here is one workaround for the problem. This query works if an outline is used to change the strategy from match to cross, since the inner cross leg contains both the contexts from the outer and inner cross legs.

```
sel a_data from T1 s
where exists (select * from T2 p
                where p.A_SEC_NAME=s.A_SEC_NAME and
                      p.a_date=s.a_date and
                      p.a_bad_cmpby =   2);
~S: Outline "TEST_BUG_OUTLINE" used
Tables:
  0 = T1
  1 = T2
  2 = T0
Cross block of 3 entries
  Cross block entry 1
    Aggregate: 0:VIA (2.A_PRICE)
    Firstn: 1
    Leaf#01 FFirst 2:T0 Card=10
      Bool: (2.A_SEC_NAME = 2.A_SEC_NAME) AND (2.A_DATE = 2.A_DATE) AND (
            2.A_PRICE_SOURCE = 'GIC') AND (2.A_PRICE_TYPE = 'STL')
      BgrNdx1 T0_NDX [2:2] Fan=9
        Keys: (2.A_PRICE_SOURCE = 'GIC') AND (2.A_PRICE_TYPE = 'STL')
        Bool: (2.A_SEC_NAME = 2.A_SEC_NAME) AND (2.A_DATE = 2.A_DATE)
  Cross block entry 2
    Leaf#02 FFirst 0:T1 Card=2
```

```
      BgrNdx1 T1_NDX [0:0] Fan=10
  Cross block entry 3
    Conjunct: <agg1> <> 0
    Aggregate-F1: 1:COUNT-ANY (<subselect>)
    Leaf#03 FFirst 1:T2 Card=2
      Bool: (1.A_SEC_NAME = 0.A_SEC_NAME) AND (1.A_DATE = 0.A_DATE) AND (CASE (
            WHEN ((1.A_CALC_PRICE > 0) AND (1.A_CALC_PRICE = <agg0>
            1.A_DET_METHOD ELSE 5) = 2)
      BgrNdx1 T2_NDX [2:2] Fan=10
        Keys: (1.A_SEC_NAME = 0.A_SEC_NAME) AND (1.A_DATE = 0.A_DATE)
-- Rdb Generated Outline : 25-JUN-2003 11:30
create outline QO_F5E5D311487F5E17_00000000
id 'F5E5D311487F5E1776847CFB5A9C308B'
mode 0
as (
  query (
-- For loop
    subquery (
      subquery (
        T0 2    access path index       T0_NDX
        )
        join by cross to
      T1 0    access path index       T1_NDX
        join by cross to
      subquery (
        T2 1    access path index       T2_NDX
        )
      )
    )
  )
compliance optional     ;
0 rows selected
```

Here is another possible workaround. The query also works if the COMPUTED BY column is changed to join the tables T0 and T2 instead of single table T0.

```
alter table T2 add column A_GOOD_CMPBY
    computed by
     case
      when (A_CALC_PRICE > 0 and
          A_CALC_PRICE =
            (select T0.A_PRICE from T0
                 where T0.A_SEC_NAME = T2.A_SEC_NAME
                     and T0.A_DATE = T2.A_DATE
                     and T0.A_PRICE_SOURCE = 'GIC'
                     and T0.A_PRICE_TYPE = 'STL' limit to 1 rows))
      then A_DET_METHOD
      else 5
    end;

sel a_data from T1 s
where exists (select * from T2 p
                where p.A_SEC_NAME=s.A_SEC_NAME and
                      p.a_date=s.a_date and
                      p.a_good_cmpby =   2);
Tables:
  0 = T1
  1 = T2
  2 = T0
Conjunct: <agg0> <> 0
```

```
Match
  Outer loop      (zig-zag)
    Get     Retrieval by index of relation 0:T1
      Index name  T1_NDX [0:0]
  Inner loop
    Aggregate: 0:COUNT-ANY (<subselect>)
    Cross block of 2 entries
      Cross block entry 1
        Get     Retrieval by index of relation 1:T2
          Index name  T2_NDX [0:0]
      Cross block entry 2
        Conjunct: CASE (WHEN ((1.A_CALC_PRICE > 0) AND (1.A_CALC_PRICE = <agg1>)
                  ) THEN 1.A_DET_METHOD ELSE 5) = 2
        Aggregate: 1:VIA (2.A_PRICE)
        Firstn: 1
        Leaf#01 FFirst 2:T0 Card=10
          Bool: (2.A_SEC_NAME = 1.A_SEC_NAME) AND (2.A_DATE = 1.A_DATE) AND (
                2.A_PRICE_SOURCE = 'GIC') AND (2.A_PRICE_TYPE = 'STL')
          BgrNdx1 T0_NDX [4:4] Fan=9
            Keys: (2.A_PRICE_SOURCE = 'GIC') AND (2.A_PRICE_TYPE = 'STL') AND (
                  2.A_SEC_NAME = 1.A_SEC_NAME) AND (2.A_DATE = 1.A_DATE)
0 rows selected
```

Notice that the conjunct with aggregate value now appears in the Cross block entry 2 under the Inner loop of the match strategy.

This query works since the context "1:T2" is joined by cross strategy with the context "2:T0" and the conjunct with aggregate value is properly resolved with all the contexts available.

This problem has been corrected in Oracle Rdb Release 7.0.7.1.

5.1.13 Bugcheck in DIO$FREE_CURRENT_LOCK for Sorted Ranked Indexes

Bug 2874671

If a cursor was used to fetch rows, and the transaction was committed between fetches, and the retrieval strategy involved a backwards scan of an index of *TYPE IS SORTED RANKED*, then Rdb could generate a bugcheck dump.

The problem occurred because Oracle Rdb mistakenly released a lock prematurely. Later, when Oracle Rdb was truly finished with the resource, a bugcheck occurred because a second attempt was made to release the same lock.

The following example uses the MF_PERSONNEL database to demonstrate the problem.

```
SQL> at 'f mf_personnel';
SQL> drop index EMP_EMPLOYEE_ID;
SQL> commit;
SQL> create unique index EMP_EMPLOYEE_ID
cont>     on EMPLOYEES (EMPLOYEE_ID asc)
cont>     type is SORTED ranked
cont>     node size 430
cont>     disable compression;
SQL> commit work;
```

```
SQL> declare transaction read write isolation level read committed;
SQL> declare t2 table cursor with hold preserve all for
cont> select    *
cont> from      employees
cont> where     employee_id > '00300'
cont> and       employee_id < '00400'
cont> order by employee_id desc;
SQL> open t2;
SQL> fetch t2;
 EMPLOYEE_ID   LAST_NAME        FIRST_NAME   MIDDLE_INITIAL
   ADDRESS_DATA_1             ADDRESS_DATA_2         CITY
      STATE   POSTAL_CODE   SEX    BIRTHDAY      STATUS_CODE
 00374         Andriola         Leslie       Q
   111 Boston Post Rd.                                Salisbury
      NH      03268         M      19-Mar-1955   1

SQL> commit;
SQL> fetch t2;
%RDMS-I-BUGCHKDMP, generating bugcheck dump file
MBRADLEY_USR:[BRADLEY]RDSBUGCHK.DMP;
```

The problem can be avoided by disabling the backward scan feature using the *RDMS$DISABLE_REVERSE_SCAN* logical name.

The problem does not occur if all rows are fetched in the same transaction.

The problem does not occur if the index being scanned is not of *TYPE IS SORTED RANKED*.

This problem has been corrected in Oracle Rdb Release 7.0.7.1.

5.1.14 Ranked Index Overflow Node Corruption on Insert

Bug 3009262

A problem in the way Rdb chooses the insertion point for the dbkey in a Ranked Index duplicate node may, in rare circumstances, cause a corruption of the index entry overflow node. This node corruption may manifest itself as a bugcheck when trying to access the records associated with the overflow node, as in the following example.

```
***** Exception at 00C72D78 : PSII2SCANGETNEXTBBCDUPLICATE + 00000128
%COSI-F-BUGCHECK, internal consistency failure
```

Or, this node corruption may be seen as a corrupt index warning when RMU/VERIFY/INDEX is used to verify the ranked index. For example:

```
%RMU-I-BTRDUPCAR, Inconsistent duplicate cardinality (C1) of 221 specified
                  for entry 1 at dbkey 85:807:1.
                  Actual count of duplicates is 251.
%RMU-I-BTRERPATH, parent B-tree node of 85:807:1 is at 85:806:0
%RMU-I-BTRROODBK, root dbkey of B-tree is 85:806:0
%RMU-W-DATNOTIDX, Row in table TT11 is not in any indexes.
                  Logical dbkey is 83:82:2.
%RMU-W-BADIDXREL, Index INDTT either points to a non-existent record or
                  has multiple pointers to a record in table TT11.
                  The logical dbkey in the index is 83:127:8.
```

A dump of the offending overflow node will show that even though the reference dbkey is correct, all the dbkeys in the overflow node may be incorrect and possibly not valid record dbkeys.

The problem may only occur on Sorted Ranked index entries that are volatile enough so that more than one overflow node is required to hold the duplicates and that, due to removal of dbkeys from the entry, at least one non−final overflow node has subsequently been removed from the entry.

A subsequent insertion of a new duplicate in that entry may, if the insertion dbkey happens to be greater than the last dbkey in an overflow node but less than the reference dbkey of the next overflow node, incorrectly update the overflow node causing the subsequent corruption.

This problem only occurs with Sorted Ranked indexes.

This problem has been corrected in Oracle Rdb Release 7.0.7.1.

5.1.15 Illegal Page Count Error in the Dynamic Optimizer

Bug 3045841

When a very large table was queried, and the access strategy used the dynamic optimizer, and the first two indexes used both returned more than 1024 dbkeys, an illegal page count error could be generated.

To encounter this error, the table had to have close to one billion (1,000,000,000) rows.

In addition, the retrieval strategy for the query must use the dynamic optimizer where at least two indices return more than 1024 dbkeys.

In this case, the dynamic optimizer will allocate memory for a dbkey bitmap. In calculating the size of that bitmap, integer (signed longword) arithmetic was used, and an integer overflow could cause the calculation to result in a negative number being used as the requested amount of memory.

The following example shows a query on a table containing one billion (1,000,000,000) rows.

```
SQL> select * from t1
cont>         where f1 >0 and f2 > 0
cont>         optimize for total time;
%COSI−F−UNEXPERR, unexpected system error
−SYSTEM−F−ILLPAGCNT, illegal page count parameter
```

A bugcheck dump may also be generated with the following exception and call sequence:

```
***** Exception at 00FE3664 : COSI_MEM_GET_VM + 000007B4
%COSI−F−UNEXPERR, unexpected system error
−SYSTEM−F−ILLPAGCNT, illegal page count parameter
Saved PC = 00FE2E68 : COSI_MEM_GET_POOL + 00000048
Saved PC = 00E3CCE8 : RDMS$$EXE_LEAF + 00001A38
Saved PC = 00E290B4 : RDMS$$EXE_OPEN + 00000764
```

The problem would not occur if any of the conditions described above were not true.

The problem can be avoided by disabling the dynamic optimizer for the query by using a query outline with the clause *EXECUTION OPTIONS NONE*.

This problem has been corrected in Oracle Rdb Release 7.0.7.1.

5.1.16 Bugcheck During Create Index with Mapping Values

Bug 2992315

SQL Create Index on a table with multiple storage areas and mapping values fails with a bugcheck.

The following example shows a typical stack trace.

```
***** Exception at 007CFA67 : PSIIBUILD$BUILD_FROM_BOTTOM + 00000693
Saved PC = 80000014 : symbol not found
Saved PC = 007CC32E : PSII$CREATE_TREE + 000000B6
Saved PC = 006AE447 : RDMS$$KOD_CREATE_TREE + 00000168
Saved PC = 006AE1AF : RDMS$$KOD_CREATE_INDEX + 00000320
Saved PC = 0061C416 : RDMS$$CREATE_INDEX_INFO + 0000234F
```

Possible workarounds include: create the index on an Alpha node or do not use mapping values.

This problem has been corrected in Oracle Rdb Release 7.0.7.1.

5.1.17 Error %RDMS−E−NOSOL_FOUND in Full Outer Join Query

Bug 2669656

The following full outer natural join query between columns of different data types fails using the cross strategy (it should succeed applying the match strategy):

```
create table x1 (f1 char(10));
create table x2 (f1 integer);
create table x3 (f1 char(10));

insert into x1 value ('1');
insert into x2 value (1);
insert into x2 value (−1);
insert into x3 value ('1');
insert into x3 value ('−1');

select * From x1 natural full outer join x2;
~S: Full OJ query with cross strategy was not possible
%RDMS−E−NOSOL_FOUND, No possible solution has been found by Rdb optimizer
```

As a workaround, the following full outer join between columns of the same data type works applying a match strategy.

```
select * From x1 natural full outer join x3;
Tables:
  0 = X1
  1 = X3
Match  (Full Outer Join)
  Outer loop
    Sort: 0.F1(a)
    Get Retrieval sequentially of relation 0:X1
```

```
  Inner loop
    Temporary relation
    Sort: 1.F1(a)
    Get Retrieval sequentially of relation 1:X3
 F1
 -1
 1
2 rows selected
```

The following query should apply a match strategy as suggested by the outline bug_outline.

```
create outline bug_outline
id '6CBC3F110B75FD48C256CE94DCEB8A1F'
mode 0
as (
  query (
-- For loop
    subquery (
      X1 0    access path sequential
        join by match to
      X2 1    access path sequential
      )
    )
  )
compliance optional     ;

select * from x1 , x2 where x1.f1 = x2.f1;
~S: Outline "BUG_OUTLINE" used
~S: Full compliance with the outline was not possible
Tables:
  0 = X1
  1 = X2
Cross block of 2 entries
  Cross block entry 1
    Get     Retrieval sequentially of relation 0:X1
  Cross block entry 2
    Conjunct: 0.F1 = 1.F1
    Get     Retrieval sequentially of relation 1:X2
 X1.F1              X2.F1
 1                      1
1 row selected
```

But the query works if one of the join predicates is cast as the same data type as the other as in the following example.

```
create outline bug_good_outline
id '0EEB4BC11CF012EDB10AEA1C16EC0E70'
mode 0
as (
  query (
-- For loop
    subquery (
      X1 0    access path sequential
        join by match to
      X2 1    access path sequential
      )
    )
  )
compliance optional     ;
```

```
select * from x1 , x2 where cast(x1.f1 as integer) = x2.f1;
~S: Outline "BUG_GOOD_OUTLINE" used
Tables:
  0 = X1
  1 = X2
Conjunct: CAST (0.F1 AS INT) = 1.F1
Match
  Outer loop
    Sort: CAST (0.F1 AS INT)(a)
    Get Retrieval sequentially of relation 0:X1
  Inner loop
    Temporary relation
    Sort: 1.F1(a)
    Get Retrieval sequentially of relation 1:X2
 X1.F1              X2.F1
 1                      1
1 row selected
```

This problem has been corrected in Oracle Rdb Release 7.0.7.1.

5.1.18 TRUNCATE TABLE and RMU /REPAIR Corruption Corrected

Previously, when using the TRUNCATE TABLE statement followed by a RMU /REPAIR /ABM command executed one or more times, various corruptions of the SPAM and ABM structures could occur. This could sometimes lead to table records that had been truncated "reappearing" in the table unexpectedly.

The following example demonstrates one possible symptom of this problem detected by RMU /VERIFY:

```
$ SQL$
    CREATE DATABASE FILENAME 'TEST.RDB'
        CREATE STORAGE AREA RDB$SYSTEM FILENAME 'RDB$SYSTEM.RDA'
        CREATE STORAGE AREA AREA FILENAME 'AREA.RDA';
    CREATE TABLE TAB (ID INTEGER);
    CREATE INDEX IND ON TAB (ID) STORE IN AREA;
    CREATE STORAGE MAP TM FOR TAB STORE IN AREA;
    COMMIT;
    INSERT INTO TAB VALUES (1);
1 row inserted
    COMMIT;
    TRUNCATE TABLE TAB;
    COMMIT;
    EXIT;
$!
$ RMU/VER/ALL/NOLOG TEST
%RMU-I-NODATANDX, no data records in index IND
$!
$ RMU/REPAIR/ABM TEST
%RMU-I-FULBACREQ, A full backup of this database should be
performed after RMU REPAIR
$ RMU/VER/ALL/NOLOG TEST
%RMU-W-AIPLAREID, area inventory page 152 entry #9 contains a
                  reference to logical area 58 that is nonexistent
%RMU-W-BADABMPTR, invalid larea for ABM page 5 in storage area 2.
                  The SPAM page entry for this page is for a different larea.
                  SPAM larea_dbid : 0 page larea_dbid: 58.
%RMU-W-BADABMPTR, invalid larea for ABM page 6 in storage area 2.
                  The SPAM page entry for this page is for a different larea.
```

```
                  SPAM larea_dbid : 0 page larea_dbid: 58.
%RMU-W-BADABMPTR, invalid larea for ABM page 7 in storage area 2.
                  The SPAM page entry for this page is for a different larea.
                  SPAM larea_dbid : 0 page larea_dbid: 58.
%RMU-E-BADABMPAG,       error verifying ABM pages
%RMU-I-NODATANDX, no data records in index IND
```

This problem has been corrected in Oracle Rdb Release 7.0.7.1. The RMU /REPAIR /ABM command no longer incorrectly updates the SPAM and ABM structures after a TRUNCATE TABLE operation.

5.1.19 UNION Query With Two Left Outer Joins in First Leg Returns Wrong Results

Bugs 3076004 and 2529598

The following UNION query with left outer join should return 1 row.

```
set flags 'strategy,detail';
select routing_id from
     (select
     C4.ROUTING_ID,
        C2.FY,
        C2.PAY_PERIOD
          from
              ROSTER as C2
                left outer join
                    WORK_AUTH as C4
                        on (C2.AUTH_NBR = C4.AUTH_NBR)
                            left outer join
                                TAX_BEN as C5
                                    ON (C2.AUTH_NBR = C5.TAX_BEN_NBR)
    union
      select
        C6.ROUTING_ID,
        C7.FY,
        C7.PAY_PERIOD
          from
              WORK_AUTH as C6, PAY_PERIOD as C7)
              AS DT (ROUTING_ID, FY, PAY_PERIOD)
    where
    fy='2004' and pay_period='01' and routing_id = 'R91297';
Tables:
  0 = ROSTER
  1 = WORK_AUTH
  2 = TAX_BEN
  3 = WORK_AUTH
  4 = PAY_PERIOD
Merge of 1 entries
  Merge block entry 1
  Reduce: <mapped field>, <mapped field>, <mapped field>
  Sort: <mapped field>(a), <mapped field>(a), <mapped field>(a)
  Merge of 2 entries
    Merge block entry 1
    Cross block of 2 entries (Left Outer Join)
      Cross block entry 1
        Cross block of 2 entries (Left Outer Join)
          Cross block entry 1
            Leaf#01 BgrOnly 0:ROSTER Card=2
```

```
            BgrNdx1 ROSTER_NDX [2:2] Fan=15
              Keys: (<mapped field> = '2004') AND (<mapped field> = '01')
        Cross block entry 2
          Leaf#02 BgrOnly 1:WORK_AUTH Card=1
            Bool: 0.AUTH_NBR = 1.AUTH_NBR
            BgrNdx1 WORK_AUTH_NDX [1:1] Fan=16
              Keys: <mapped field> = 'R91297'
    Cross block entry 2
      Conjunct: 0.AUTH_NBR = 2.TAX_BEN_NBR
      Index only retrieval of relation 2:TAX_BEN
        Index name  TAX_BEN_NDX [1:1]
          Keys: 0.AUTH_NBR = 2.TAX_BEN_NBR
  Merge block entry 2
  Cross block of 2 entries
    Cross block entry 1
      Conjunct: 3.ROUTING_ID = 'R91297'
      Index only retrieval of relation 3:WORK_AUTH
        Index name  WORK_AUTH_NDX [1:1]
          Keys: <mapped field> = 'R91297'
    Cross block entry 2
      Conjunct: (4.FY = '2004') AND (4.PAY_PERIOD = '01')
      Index only retrieval of relation 4:PAY_PERIOD
        Index name  PAY_PERIOD_NDX [2:2]
          Keys: (<mapped field> = '2004') AND (<mapped field> = '01')
 ROUTING_ID
 R91297
 NULL
2 rows selected
```

This problem is similar to the previous Bug 2529598 where the fix did not cover the current case with two left outer joins in the first UNION leg. The conjunct "routing_id = 'R91297'" is generated at the top of the second UNION (merge) leg but not at the top of the first leg and thus returns the wrong result.

The query works if the legs of the UNION clause are swapped, as in the following example:

```
select routing_id from
     (
      select
        C6.ROUTING_ID,
        C7.FY,
        C7.PAY_PERIOD
          from
              WORK_AUTH as C6, PAY_PERIOD as C7
    union
      select
      C4.ROUTING_ID,
        C2.FY,
        C2.PAY_PERIOD
          from
              ROSTER as C2
                left outer join
                    WORK_AUTH as C4
                        on (C2.AUTH_NBR = C4.AUTH_NBR)
                            left outer join
                                TAX_BEN as C5
                                    ON (C2.AUTH_NBR = C5.TAX_BEN_NBR)
                                    )
                                    AS DT (ROUTING_ID, FY, PAY_PERIOD)
    where
    fy='2004' and pay_period='01' and routing_id = 'R91297';
```

```
Tables:
  0 = WORK_AUTH
  1 = PAY_PERIOD
  2 = ROSTER
  3 = WORK_AUTH
  4 = TAX_BEN
Conjunct: <mapped field> = 'R91297'
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: 1.FY = '2004'
  Conjunct: 1.PAY_PERIOD = '01'
  Merge of 2 entries
    Merge block entry 1
    Cross block of 2 entries
      Cross block entry 1
        Conjunct: 0.ROUTING_ID = 'R91297'
        Index only retrieval of relation 0:WORK_AUTH
          Index name  WORK_AUTH_NDX [1:1]
            Keys: <mapped field> = 'R91297'
      Cross block entry 2
        Conjunct: (1.FY = '2004') AND (1.PAY_PERIOD = '01')
        Index only retrieval of relation 1:PAY_PERIOD
          Index name  PAY_PERIOD_NDX [2:2]
            Keys: (<mapped field> = '2004') AND (<mapped field> = '01')
    Merge block entry 2
    Cross block of 2 entries        (Left Outer Join)
      Cross block entry 1
        Cross block of 2 entries        (Left Outer Join)
          Cross block entry 1
            Leaf#01 BgrOnly 2:ROSTER Card=2
              BgrNdx1 ROSTER_NDX [2:2] Fan=15
                Keys: (<mapped field> = '2004') AND (<mapped field> = '01')
          Cross block entry 2
            Conjunct: 2.AUTH_NBR = 3.AUTH_NBR
            Get     Retrieval by index of relation 3:WORK_AUTH
              Index name  WORK_AUTH_NDX [0:0]
      Cross block entry 2
        Leaf#02 BgrOnly 4:TAX_BEN Card=1
          Bool: 2.AUTH_NBR = 4.TAX_BEN_NBR
          BgrNdx1 TAX_BEN_NDX [1:1] Fan=17
            Keys: 2.AUTH_NBR = 4.TAX_BEN_NBR
 ROUTING_ID
 R91297
1 row selected
```

Notice that the "Conjunct: <mapped field> = 'R91297'" is now generated on top of both UNION legs. Even though the conjunct is NOT efficiently optimized by being pushed down to the second leg, at least the query works correctly.

There is no known workaround other than the above−mentioned query with UNION legs swapped.

This problem has been corrected in Oracle Rdb Release 7.0.7.1.

5.1.20 Logical Area Record Erasure Count Not Updated for Cached Rows

Bug 3099718

Previously, logical area statistics for record erase operations were not correctly counted when erasing rows in a row cache.

This problem has been corrected in Oracle Rdb Release 7.0.7.1.

5.1.21 Query With Sum Function of Two Select Counts Bugchecks

Bug 2649215

A query with a sum function of two select counts could produce a bugcheck.

```
    select sum ((select count (*) - (select count (*) from T1
                                  where T1.F1 = T2.F1
                                    and T1.F2 = T2.F2)
              from T2
              where F3 = 'B'
              group by T2.F1, T2.F2)
              )
    from rdb$database;
%DEBUG-I-DYNMODSET, setting module RDMS$PREEXEASN
%SYSTEM-F-BREAK, breakpoint fault at PC=003995E9, PSL=03C00004
break on exception at RDMS$PREEXEASN\RDMS$$FIND_VALID_SEG_CRTV\%LINE 8443 in
EAD 1
```

The query works if one of the equality predicates is removed, as in the following example.

```
select sum ((select count (*)
                - (select count (*) from T1 where
                     T1.F1 = T2.F1
!---                 and T1.F2 = T2.F2
                    )
              from T2 where F3 = 'B'
              group by T2.F1, T2.F2
              ))
from rdb$database;
Tables:
  0 = RDB$DATABASE
  1 = T2
  2 = T1
Aggregate: 0:SUM (<agg1>)
Cross block of 2 entries
  Cross block entry 1
    Aggregate: 1:VIA (<mapped field> - <agg2>)
    Cross block of 2 entries
      Cross block entry 1
        Aggregate: 2:COUNT (*)
        Index only retrieval of relation 2:T1
          Index name  U1_T1 [1:1]
            Keys: 2.F1 = 1.F1
```

```
      Cross block entry 2
        Aggregate: 3:COUNT (*)
        Conjunct: 1.F3 = 'B'
        Get     Retrieval by index of relation 1:T2
          Index name  U1_T2 [0:0]
  Cross block entry 2
    Retrieval sequentially of relation 0:RDB$DATABASE

                    1
1 row selected
```

The query also works if the SUM function is removed.

```
select count (*) − (select count (*) from T1
                                  where T1.F1 = T2.F1
                                    and T1.F2 = T2.F2)
             from T2
             where F3 = 'B'
             group by T2.F1, T2.F2
             ;
Tables:
  0 = T2
  1 = T1
Cross block of 2 entries
  Cross block entry 1
    Aggregate: 0:COUNT (*)
    Conjunct: 0.F3 = 'B'
    Get     Retrieval by index of relation 0:T2
      Index name  U1_T2 [0:0]
  Cross block entry 2
    Aggregate: 1:COUNT (*)
    Index only retrieval of relation 1:T1
      Index name  U1_T1 [2:2]        Direct lookup
        Keys: (1.F1 = 0.F1) AND (1.F2 = 0.F2)

                    0
1 row selected
```

There is no known workaround for this problem.

This problem has been corrected in Oracle Rdb Release 7.0.7.1.

5.1.22 Left Outer Join Query With CONCAT Function Returns Wrong Results

Bugs 3139961, 2836144, 1837522

The following query with CONCAT function returns wrong results (should return 4 rows).

```
set flags 'strategy,detail';
select T1.YEAR,T1.WEEK,T1.KODE
  FROM  T1 LEFT OUTER JOIN T2
          ON T1.YEAR = T2.YEAR AND T1.KODE = T2.KODE,
        T3 where
    (T1.YEAR||T1.WEEK <='200337') and
    T3.IPROC = T1.IPROC AND
    T3.ITEAM = T1.ITEAM     ;
```

```
Tables:
  0 = T1
  1 = T2
  2 = T3
Cross block of 2 entries
  Cross block entry 1
    Index only retrieval of relation 2:T3
      Index name  T3_IND1 [0:0]
  Cross block entry 2
    Conjunct: ((0.YEAR < SUBSTRING ('200337' FROM 0 FOR 4)) AND
              <error: missing expression>) OR ((0.YEAR = SUBSTRING ('200337'
               FROM 0 FOR 4)) AND (0.WEEK <= SUBSTRING ('200337' FROM 4)))
    Cross block of 2 entries        (Left Outer Join)
      Cross block entry 1
        Leaf#01 FFirst 0:T1 Card=4
          Bool: (((0.YEAR < SUBSTRING ('200337' FROM 0 FOR 4)) AND NOT
                MISSING (0.WEEK)) OR ((0.YEAR = SUBSTRING ('200337'
                 FROM 0 FOR 4)) AND (0.WEEK <= SUBSTRING ('200337' FROM 4)))
                ) AND (2.IPROC = 0.IPROC) AND (2.ITEAM = 0.ITEAM)
          BgrNdx1 T1_IND2 [0:0] Fan=15
          BgrNdx2 T1_IND1 [0:0] Fan=14
            Bool: ((0.YEAR < SUBSTRING ('200337' FROM 0 FOR 4)) AND NOT
                  MISSING (0.WEEK)) OR ((0.YEAR = SUBSTRING ('200337'
                   FROM 0 FOR 4)) AND (0.WEEK <= SUBSTRING ('200337' FROM 4)
                  ))
      Cross block entry 2
        Conjunct: (0.YEAR = 1.YEAR) AND (0.KODE = 1.KODE)
        Index only retrieval of relation 1:T2
          Index name  T2_IND1 [2:2]      Direct lookup
            Keys: (0.YEAR = 1.YEAR) AND (0.KODE = 1.KODE)
 T1.YEAR   T1.WEEK   T1.KODE
 2003      26        270
 2003      34        270
2 rows selected
```

The problem is caused by the fix made for Bug 1837522 in Oracle Rdb Release 7.0.6.2 where a left outer join query with OR predicate returns wrong results.

Even though this query apparently does not contain an OR predicate, the Oracle Rdb optimizer transforms the CONCAT function into an OR expression with two SUBSTRING functions, as seen in the following detail strategy.

```
  Bool: (((0.YEAR < SUBSTRING ('200337' FROM 0 FOR 4)) AND
          NOT MISSING (0.WEEK)) OR
         ((0.YEAR = SUBSTRING ('200337' FROM 0 FOR 4)) AND
          (0.WEEK <= SUBSTRING ('200337' FROM 4)))) AND
        (2.IPROC = 0.IPROC) AND (2.ITEAM = 0.ITEAM)
```

There is no known workaround for this problem.

This problem has been corrected in Oracle Rdb Release 7.0.7.1.

5.1.23 RCS Bugchecks at DIOCCH$UNMARK_GRIC_ENT

Bug 2502144

In prior releases of Oracle Rdb, it was possible for the Record Cache Server (RCS) process to bugcheck in DIOCCH$UNMARK_GRIC_ENT. This problem was due to an incorrect check in the RCS process while evaluating the amount of locked space on a database page that is owned by the RCS.

When the RCS process writes an erased or resized record back to the database page, it must return the resultant locked space on the page back to free space. This is because the RCS is not allowed to "own" locked space. As part of this action, the RCS was checking to make sure that the amount of locked space being returned exactly matched the length of the space being returned by the erased or resized record. This check was not taking into account the possibility that there was existing locked space on the page marked with the TID of the RCS. In such a case, the RCS could find that it "owned" more locked space than it expected and would bugcheck.

This problem has been corrected in Oracle Rdb Release 7.0.7.1. The RCS process now correctly evaluates and releases all locked space on the page that is owned by the RCS after writing an erased or resized record back to the database.

5.1.24 Wrong Sort Order for Query with Aggregate, Group By, Order By

Bug 3077857

Before the problem was fixed, the query in the example below returned the results in ascending order of the NTS_FLAG column, in contradiction to the explicit DESCENDING attribute in the ORDER BY clause. The problem appeared when RDM$BIND_BUFFERS was set to 200 but not when it was set to 20.

```
set flags 'detail,strategy';

select  Max(SERVICE_SUM.NTS_FLAG) NTS_FLAG
  from  SERVICE_SUM, INFO_PROVIDER_CODE_PARENT
  where SERVICE_SUM.CALL_END_DATE between
        DATE ANSI '2003-07-16' and  DATE ANSI '2003-07-16'
        and
        SERVICE_SUM.DIALED_NO = INFO_PROVIDER_CODE_PARENT.DIALED_NO
  group by SERVICE_SUM.NTS_FLAG
  order by SERVICE_SUM.NTS_FLAG DESC;
```

The change in the value for RDMS$BIND_BUFFERS had the effect of changing the optimizer query strategy from a CROSS join to a MATCH join. Using interactive SQL, for example, this could be seen by first enabling output of the optimizer strategy (see the SET FLAGS statement in the preceding example). The combination of MATCH join, aggregate query, ascending GROUP BY sort, and descending ORDER BY sort, plus the optimizer logic to try to eliminate unnecessary sorts were the conditions needed for the problem to appear.

As a workaround, the problem could be made to go away by reducing the value in RDM$BIND_BUFFERS from 200 to 20 or by creating a query outline forcing a CROSS join strategy.

This problem has been corrected in Oracle Rdb Release 7.0.7.1.

5.1.25 Left Outer Join Query with SUBSTRING and CHAR_LENGTH Bugchecks

Bug 2851595

The following left outer join query with SUBSTRING and CHAR_LENGTH bugchecks during the process of creating a match retrieval strategy.

```
select e.employee_id
 from employees e left outer join job_history jh
 on (e.employee_id =
     substring (jh.employee_id from 1
                 for char_length (e.employee_id)));
%DEBUG-I-DYNMODSET, setting module RDMS$PREEXEMSC
%SYSTEM-F-BREAK, breakpoint fault at PC=003A4CD6, PSL=03C00004
break on exception at RDMS$PREEXEMSC\RDMS$$FIND_MEMBER_EQV_CLASS\%LINE 4486
DBG> SH CA
 module name    routine name     line               rel PC           ab
*RDMS$PREEXEMSC RDMS$$FIND_MEMBER_EQV_CLASS
*RDMS$PREEXE    RDMS$$SETUP_SORT32
*RDMS$PREEXE_CREATE
                RDMS$$CREATE_RSS$MTCH
```

The bugcheck is caused by a match order using equi-join keys of the outer join query, created from the following predicate:

```
 (e.employee_id = substring (jh.employee_id from 1
                             for char_length (e.employee_id))
```

where the match key on the right side depends on its own context "employees e" from its left side.

As a workaround, the query works if the function CHAR_LENGTH is applied with JH table:

```
select e.employee_id
 from employees e left outer join job_history jh
 on (e.employee_id =
     substring (jh.employee_id from 1
                 for char_length (jh.employee_id)));
Tables:
  0 = EMPLOYEES
  1 = JOB_HISTORY
Match    (Left Outer Join)
  Outer loop
    Index only retrieval of relation 0:EMPLOYEES
      Index name  EMP_EMPLOYEE_ID [0:0]
  Inner loop
    Temporary relation
    Sort: SUBSTRING (1.EMPLOYEE_ID FROM (1 - 1) FOR CHAR_LENGTH (1.EMPLOYEE_ID))
          (a)
    Index only retrieval of relation 1:JOB_HISTORY
      Index name  JH_EMPLOYEE_ID [0:0]
 E.EMPLOYEE_ID
 00164
 00164
 ...etc...

274 rows selected
```

This problem has been corrected in Oracle Rdb Release 7.0.7.1.

5.1.26 Join Query with GROUP_BY/ORDER_BY Returns Wrong Order

Bug 2851708

The following query with GROUP_BY/ORDER_BY, joining two derived tables with similar GROUP_BY clauses, returns results in the wrong order (should return in descending order).

```
set flags 'strategy, detail';
select  t1.name, t2.name, t1.datum, t2.datum
  from (select name, datum from a
          group by name, datum) t1
       join
       (select name, datum from b
          group by name, datum) t2
       on (t1.name = t2.name and t1.datum = t2.datum)
   group by t1.name, t2.name, t1.datum, t2.datum
   order by t1.name desc, t1.datum desc ;
Tables:
  0 = A
  1 = B
Reduce: 0.NAME, 0.DATUM, 1.NAME, 1.DATUM
Cross block of 2 entries
  Cross block entry 1
    Merge of 1 entries
      Merge block entry 1
      Reduce: 0.NAME, 0.DATUM
      Sort: 0.NAME(a), 0.DATUM(a)           <== See Note:
      Get Retrieval sequentially of relation 0:A
  Cross block entry 2
    Merge of 1 entries
      Merge block entry 1
      Reduce: 1.NAME, 1.DATUM
      Sort: 1.NAME(d), 1.DATUM(d)
      Conjunct: (0.NAME = 1.NAME) AND (0.DATUM = 1.DATUM)
      Get     Retrieval sequentially of relation 1:B
 T1.NAME   T1.DATUM
 AAAA      1-JAN-2000 00:00:00.00
 BBBB      1-JAN-2000 00:00:00.00
2 rows selected
```

NOTE: The sort keys should be descending instead of ascending.

This problem occurs when the main query contains a GROUP BY clause on the columns of the two joined derived tables with GROUP BY with the ORDER BY clause referencing the columns of the first table using descending order.

There is no known workaround for this problem.

This problem has been corrected in Oracle Rdb Release 7.0.7.1.

5.1.27 Query Joining Two Derived Tables of a View with UNION Overflows the Stack

Bug 3194445

The following query, joining two derived tables of a view with UNION, overflows the stack.

```
select
  (select period_start_date from gl_period_vw
     where
       company_no = glpv.company_no and
       fiscal_yr = glpv.fiscal_yr
        )
     as p2_period_start_date,
  (select period_end_date from gl_period_vw
     where
       company_no = glpv.company_no and
       fiscal_yr = glpv.fiscal_yr
        )
     as p2_period_end_date
from gl_period_vw glpv, gl_period glp
;
%RDB-F-IMP_EXC, facility-specific limit exceeded
-RDB-I-TEXT, internal error -- query solution not found
```

The view is defined as:

```
create view GL_PERIOD_VW
    (COMPANY_NO,
     FISCAL_YR,
     PERIOD_START_DATE,
     PERIOD_END_DATE) as
    select
        C2.COMPANY_NO,
        C2.FISCAL_YR,
        C2.PERIOD_START_DATE,
        C2.PERIOD_END_DATE
    from GL_PERIOD C2
        where (C2.PERIOD_NO = 0)
    union all
    select
        C3.COMPANY_NO,
        C3.FISCAL_YR,
        C3.PERIOD_START_DATE,
        C3.PERIOD_END_DATE
    from GL_PERIOD C3, INTEGER_LIST C4
        where (C3.PERIOD_NO = 0)
    ;
```

This problem is caused by the fix made for Bug 2649215 where a query before bugchecks.

There is no known workaround for this problem.

This problem has been corrected in Oracle Rdb Release 7.0.7.1.

5.1.28 Query with Shared Expression in Two Predicates Returns Wrong Results

Bug 3201864

The following query with shared expression in two predicates should return 1 row but instead returns 2 rows.

```
set flags 'strategy,detail';

SELECT  T1.CODE,T2.ALUM, T1.ALUM, T2.CODE
  FROM T1, T2, T3 WHERE
          T2.TUNE    = T1.TUNE          AND
          T2.SLAM    = T1.SLAM          AND
          T2.STAR    = T1.STAR          AND
          (T1.CODE  >= T2.ALUM          AND
           (T1.ALUM <= T2.CODE OR T2.CODE IS NULL)) AND
          T3.TUNE    = T1.TUNE          AND
          T3.SLAM    = T1.SLAM          AND
          T3.STAR    = T1.STAR          AND
          T2.TUNE    = '240167-1447'    AND
          T2.SLAM    <= '31-dec-2002'   AND
          (T2.CODE   >= '01-jan-2002' OR T2.CODE  IS NULL) AND
         NOT EXISTS
            (SELECT T4.TUNE FROM T2 T4 WHERE
                  T4.TUNE = T2.TUNE    AND
                  T4.MAKE = '0'       AND
                  T4.SLAM = T2.SLAM    AND
                  T4.STAR    = T2.STAR AND
                  T4.SITE <> T2.SITE AND
                  T1.ALUM >= T4.ALUM AND
                  T1.CODE <= T4.CODE );
Tables:
  0 = T1
  1 = T2
  2 = T3
  3 = T2
Cross block of 4 entries
  Cross block entry 1
    Conjunct: 1.TUNE = '240167-1447'
    Conjunct: 1.SLAM <= '31-DEC-2002'
    Leaf#01 FFirst 1:T2 Card=2
      Bool: (1.CODE >= '1-JAN-2002') OR MISSING (1.CODE)
      BgrNdx1 T2_IND_1 [1:2] Fan=7
        Keys: (1.TUNE = '240167-1447') AND (1.SLAM <= '31-DEC-2002')
  Cross block entry 2
    Leaf#02 FFirst 0:T1 Card=1
      Bool: (1.TUNE = 0.TUNE) AND (1.SLAM = 0.SLAM) AND (1.STAR = 0.STAR) AND
            (0.CODE >= 1.ALUM)
      BgrNdx1 T1_IND [1:1] Fan=11
        Keys: 1.TUNE = 0.TUNE
      BgrNdx2 T1_IND_1 [3:3] Fan=8
        Keys: (1.TUNE = 0.TUNE) AND (1.SLAM = 0.SLAM) AND (1.STAR = 0.STAR)
  Cross block entry 3
    Conjunct: <agg0> = 0
    Aggregate-F1: 0:COUNT-ANY (<subselect>)
    Leaf#03 FFirst 3:T2 Card=2
      Bool: (3.TUNE = 1.TUNE) AND (3.MAKE = '0') AND (3.SLAM = 1.SLAM) AND
            (3.STAR = 1.STAR) AND (3.SITE <> 1.SITE) AND (0.ALUM >= 3.ALUM)
            AND (0.CODE <= 3.CODE)
```

```
      BgrNdx1 T2_IND_1 [3:4] Fan=7
        Keys: (3.TUNE = 1.TUNE) AND (3.SLAM = 1.SLAM) AND
              (3.STAR = 1.STAR) AND (0.ALUM >= 3.ALUM)
        Bool: (3.SLAM <= '31-DEC-2002') AND (3.TUNE = '240167-1447')
  Cross block entry 4
    Index only retrieval of relation 2:T3
      Index name  T3_IND [3:3]
        Keys: (2.TUNE = 0.TUNE) AND (2.SLAM = 0.SLAM) AND (2.STAR = 0.STAR)
 T1.CODE                    T2.ALUM               T1.ALUM
   T2.CODE
 22-FEB-2002 00:00:00.00    1-OCT-2001 00:00:00.00   22-FEB-2002 00:00:00.00
   31-JAN-2002 00:00:00.00

 22-FEB-2002 00:00:00.00    1-FEB-2002 00:00:00.00   22-FEB-2002 00:00:00.00
   31-DEC-2003 00:00:00.00
2 rows selected
```

The predicate "T2.CODE IS NULL" is shared by two OR clauses referencing other different tables as in the following example.

```
   the first predicate (T1.ALUM <= T2.CODE OR T2.CODE  IS NULL)) references
table T1 and T2, and
   the second predicate (T2.CODE  >= '01-jan-2002' OR T2.CODE  IS NULL)
references table T2.
```

However, in the detailed strategy display, the first predicate is missing under the cross block entry 2 for table "0:T1".

As a workaround, the query works if the SQL flag 'MAX_STABILITY' is defined OR the logical name RDMS$MAX_STABILITY is defined.

```
set flags 'max_stability';

Tables:
  0 = T1
  1 = T2
  2 = T3
  3 = T2
Cross block of 4 entries
  Cross block entry 1
    Conjunct: (1.CODE >= '1-JAN-2002') OR MISSING (1.CODE)
    Get     Retrieval by index of relation 1:T2
      Index name  T2_IND_1 [1:2]
        Keys: (1.TUNE = '240167-1447') AND (1.SLAM <= '31-DEC-2002')
  Cross block entry 2
    Conjunct: (1.SLAM = 0.ALUM) AND (1.STAR = 0.STAR) AND (0.CODE >= 1.ALUM)
              AND ((0.ALUM <= 1.CODE) OR MISSING (1.CODE))
    Get     Retrieval by index of relation 0:T1
      Index name  T1_IND [1:1]
        Keys: 1.TUNE = 0.TUNE
  Cross block entry 3
    Conjunct: <agg0> = 0
    Aggregate-F1: 0:COUNT-ANY (<subselect>)
    Conjunct: (3.MAKER = '0') AND (3.SITE <> 1.SITE) AND (0.CODE <= 3.CODE)
    Get     Retrieval by index of relation 3:T2
      Index name  T2_IND_1 [3:4]
        Keys: (3.TUNE = 1.TUNE) AND (3.SLAM = 1.SLAM) AND
              (3.STAR = 1.STAR) AND (0.ALUM >= 3.ALUM)
```

```
        Bool: (3.SLAM <= '31-DEC-2002') AND (3.TUNE = '240167-1447')
  Cross block entry 4
    Index only retrieval of relation 2:T3
      Index name  T3_IND [3:3]
        Keys: (2.TUNE = 0.TUNE) AND (2.ALUM = 0.ALUM) AND (2.STAR = 0.STAR)
 T1.CODE                    T2.ALUM                   T1.ALUM
   T2.CODE
 22-FEB-2002 00:00:00.00    1-FEB-2002 00:00:00.00    22-FEB-2002 00:00:00.00
   31-DEC-2003 00:00:00.00

1 row selected
```

This problem has been corrected in Oracle Rdb Release 7.0.7.1.

5.1.29 Wrong Index Retrieval is Selected in Query with GTR Predicate

Bug 3144382

The wrong index retrieval is selected in a query with a GTR predicate.

```
set flags 'strategy,detail';

create index t_i on t(v1,v2,v3,v4,v5);

select * from t where
    v1='D' and v2 > 18 and
    v3 > 7 and v4 > 17 and v5 > 4
    order by v1,v2,v3,v4,v5;
Tables:
  0 = T
Conjunct: (0.V1 = 'D') AND (0.V2 > 18) AND (0.V3 > 7) AND (0.V4 > 17) AND (0.V5
           > 4)
Get     Retrieval by index of relation 0:T
  Index name  T_I [1:1]
    Keys: 0.V1 = 'D'
    Bool: (0.V2 > 18) AND (0.V3 > 7) AND (0.V4 > 17) AND (0.V5 > 4)
...etc...
180 rows selected
```

The query takes about 10.3 seconds (elapsed time) on a VAX machine.

In Oracle Rdb Release 7.0−62, the query runs fast (approximately 1 second) with the following strategy:

```
Tables:
  0 = T
Conjunct: ((0.V1 = <cvar>) AND (0.V2 >
<cvar>) AND (0.V3 > <cvar>)
    AND (0.V4 > <cvar>) AND (0.V5 > <cvar>))
Get     Retrieval by index of relation 0:T
  Index name  T_I [2:1] Bool
    Key: (0.V1 = <cvar>) AND (0.V2 > <cvar>)
    Bool: (0.V3 > <cvar>) AND (0.V4 > <cvar>) AND (0.V5 > <cvar>)
```

As a workaround, the query works if the SQL flag 'NOMAX_SOLUTION' is defined OR the logical name RDMS$DISABLE_MAX_SOLUTION is defined.

```
set flags 'NOMAX_SOLUTION';

select * from t where
    v1='D' and v2 > 18 and
    v3 > 7 and v4 > 17 and v5 > 4
    order by v1,v2,v3,v4,v5;
Tables:
  0 = T
Conjunct: (0.V1 = 'D') AND (0.V2 > 18) AND (0.V3 > 7) AND (0.V4 > 17) AND (0.V5
          > 4)
Get     Retrieval by index of relation 0:T
  Index name  T_I [2:1]
    Keys: (0.V1 = 'D') AND (0.V2 > 18)
    Bool: (0.V3 > 7) AND (0.V4 > 17) AND (0.V5 > 4)
```

The query takes about 2.22 seconds (elapse time) on a VAX machine.

This problem has been corrected in Oracle Rdb Release 7.0.7.1.

5.1.30 Memory Corruption When Using Explicit 2PC

Bug 3230612

It was possible for an application to experience memory corruption when the following conditions were true:

- Explicit two−phase commit (2PC) transactions were being utilized. That is, transactions that were started and ended by using the system services SYS$START_TRANS and SYS$END_TRANS.
- The application would:
    1. Detach from the database involved in the 2PC transaction
    2. Attach to a database
    3. Start a new 2PC transaction
- The application would allocate memory after detaching from the database and before attaching to a database.

  This problem can be avoided by not using explicit two−phase commit or by remaining attached to all databases.

  For more information regarding explicit two−phase commit transactions, see the Guide to Distributed Transactions.

  This problem has been corrected in Oracle Rdb Release 7.0.7.1.

5.1.31 Query Applying Zero Shortcut Returns Wrong Results

Bug 3216607

The following query applying zero shortcut optimization should return 7 rows.

```
set flags 'strategy,detail';
SQL> select wip_number, plan_sequence_code
cont> from pv_errors
cont> where wip_number = 'PML_257224' and plan_sequence_code = 1;
```

```
~S#0004
Leaf#01 FFirst PV_ERRORS Card=151726
  BgrNdx1 PV_ERRORS_HASH [1:1] Fan=1
  BgrNdx2 PV_ERRORS_SORTED_001 [2:2] Fan=13
~E#0004.01(1) Estim   Ndx:Lev/Seps/DBKeys 1:_6 2:1/0/0 ZeroShortcut
0 rows selected
```

A number of optimization changes have been made to the latest releases of Rdb to improve how the optimizer chooses which indexes should be used to help retrieve data.

One such improvement has been new index estimation procedures that give much more precise estimation of the effectiveness of the use of a particular index for retrieval.

This problem may occur when the optimizer incorrectly determines that one of the indices contains no records that could possibly satisfy the selection criteria and therefore immediately discontinues the processing of the query, delivering no records.

This is called 'zero shortcut' optimization which can be seen in the execution trace of the query above.

```
~E#0004.01(1) Estim   Ndx:Lev/Seps/DBKeys 1:_6 2:1/0/0 ZeroShortcut
```

As a workaround for the problem, the query returns the correct result by adding an ORDER BY clause.

```
SQL> select wip_number, plan_sequence_code
cont> from pv_errors
cont> where wip_number = 'PML_257224' and plan_sequence_code = 1
cont> order by wip_number;
 WIP_NUMBER       PLAN_SEQUENCE_CODE
  PML_257224                       1
  PML_257224                       1
  PML_257224                       1
  PML_257224                       1
  PML_257224                       1
  PML_257224                       1
  PML_257224                       1
~S#0003
Leaf#01 Sorted PV_ERRORS Card=151726
  FgrNdx  PV_ERRORS_HASH [1:1] Fan=1
  BgrNdx1 PV_ERRORS_SORTED_001 [2:2] Fan=13
 WIP_NUMBER       PLAN_SEQUENCE_CODE
 PML_257224                       1
 PML_257224                       1
 PML_257224                       1
 PML_257224                       1
 PML_257224                       1
 PML_257224                       1
~E#0003.01(1) FgrNdx  Sorted   DBKeys=8  Fetches=0+0  RecsOut=7
~E#0003.01(1) FgrNdx  Sorted   DBKeys=8  Fetches=0+0  RecsOut=7
 PML_257224                       1
7 rows selected
```

This problem has been corrected in Oracle Rdb Release 7.0.7.1.

5.2 SQL Errors Fixed

5.2.1 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 clauses, 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 problem has been corrected in Oracle Rdb Release 7.0.7.1.

5.2.2 Incorrect Unit for the DETECTED ASYNC PREFETCH THRESHOLD Option

Bug 2838771

In prior versions of Oracle Rdb V7.0, the THRESHOLD option of the DETECTED ASYNC PREFETCH clause required the units to be PAGES. However, this value is really specified in BUFFERS.

To avoid confusion, the SQL syntax has now been changed to use the BUFFERS unit for this clause. The older syntax is now deprecated as shown in the following example:

```
SQL> alter database
cont>     filename 'DB$:PERSONNEL'
cont>     detected async prefetch is ENABLED
cont>         (depth is 4 buffers, threshold is 4 pages);
%SQL-I-DEPR_FEATURE, Deprecated Feature:
PAGES is replaced with BUFFERS
```

This problem has been corrected in Oracle Rdb Release 7.0.7.1. In addition, the RMU Extract command will output the new corrected syntax.

```
SQL> alter database
```

```
cont>     filename 'DB$:PERSONNEL'
cont>     detected async prefetch is ENABLED
cont>         (depth is 4 buffers, threshold is 4 buffers);
```

5.2.3 DECLARE LOCAL TEMPORARY TABLE Limited to 10 Tables Per Session

Bug 2911428

In prior releases of Oracle Rdb, access to tables declared using the DECLARE LOCAL TEMPORARY TABLE statement in interactive and dynamic SQL would fail. This problem does not occur when DECLARE LOCAL TEMPORARY TABLE is used in a CREATE MODULE statement.

The following example shows the reported error.

```
SQL> select * from demo.MODULE.prodn_new_constraints_table;
%RDB-E-OBSOLETE_METADA, request references metadata objects that no longer exist
-RDMS-F-TABIDNOTDEF, relation ID, 11, is not defined in database
```

This problem has been corrected in Oracle Rdb Release 7.0.7.1. This limitation has been removed from interactive and dynamic SQL. A workaround would be to declare tables #10 and #11 and not use those temporary tables.

5.2.4 IVP or Other Failure with Dynamic SQL if SQL$INT is Installed /RESIDENT

Bug 2950983

In SYS$STARTUP:SQL$STARTUP.COM, if the line RESIDENT = "/RESIDENT" was present, (for example, not commented out), the IVP failed while running the dynamic SQL test.

This problem has been corrected in Oracle Rdb Release 7.0.7.1.

5.2.5 Bugcheck at PSIINDEX$FIND_ENTS_EXACT + 54

Bug 2448304

When a combination of metadata operations was being performed in the same transaction, it was possible that an OpenVMS Alpha bugcheck could be generated with an exception similar to the following example.

```
***** Exception at 0106BC24 : PSIINDEX$FIND_ENTS_EXACT + 00000054
%SYSTEM-F-ACCVIO, access violation, reason mask=00,
  virtual address=0000000050000038, PC=000000000106BC24, PS=0000000B
```

This problem was caused by an incorrect use of internal memory management optimizations. This incorrect use can be identified by the unusual virtual adress of *0000000050000038*.

The problem can be avoided by reducing the number of metadata operations performed in a single transaction.

This problem has been corrected in Oracle Rdb Release 7.0.7.1.

5.2.6 Multistatement Procedures Used with Connections Resulted in %RDB−E−OBSOLETE_METADA Error Message

Bugs 1879521 and 1808821

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

Note: this problem was originally reported as fixed in Release 7.0.6.4 but the fix was inadvertently left out of that release.

This problem has been corrected in Oracle Rdb Release 7.0.7.1.

5.2.7 IMPORT May Generate ACCVIO Exception During Import of a Module

In previous releases of Oracle Rdb, the IMPORT command would fail with an ACCVIO exception during import of a module that contained an external function or procedure.

The following example shows the exception.

```
SQL> import database
cont>    from SQL_CREATE_MODULE_4G_EXP
cont>    filename SQL_CREATE_MODULE_4G_DB
cont> ;
%SYSTEM−F−ACCVIO, access violation, reason mask=00,
virtual address=00000000, PC=0029908E, PSL=03C00001
```

This problem has been corrected in Oracle Rdb Release 7.0.7.1. SQL IMPORT now correctly handles these types of modules.

5.2.8 Unexpected ACCVIO When Reporting Incompatible Character Set Assignments

Bug 3098432

In prior releases of Rdb, an incompatible character set assignment might cause Rdb to generate a malformed message vector which can lead to an ACCVIO while trying to display the message.

The following example shows the problem.

```
SQL> select count(*) from rdb$database
cont>  where _dec_mcs'a'=translate('a' using rdb$dec_kanji);
%RDB-E, invalid or unsupported data conversion
%SYSTEM-F-ACCVIO, access violation, reason mask=00, virtual address=
00000000004F2DE4, PC=FFFFFFFF80207624, PS=0000001B
```

This problem has been corrected in Oracle Rdb Release 7.0.7.1.

5.2.9 SQL−F−NODBFILE When SQL Modules are Compiled With /CONNECT

Bug 3002999

When multiple SQL modules were called from the same main program and a disconnect was done, under some circumstances calls to SQL modules after the disconnect would fail with the following message:

```
%SQL-F-NODBFILE, Alias <ALIAS_NAME> is missing a declaration
```

where <ALIAS_NAME> is one of the aliases declared by one of the SQL modules. The behaviour will occur whenever the following is true:

- The SQL modules are compiled with /CONNECT (which is the default).
- Some aliases have a run time resolution for the filename or pathname.
- One or more modules do not declare one of the aliases which has a run−time resolution.
- The first call after the disconnect is to a module which does not have a declaration of an alias with a run−time resolution.

As a workaround, ensure that the first SQL module called after the disconnect declares all aliases which have a run−time resolution.

This problem has been corrected in Oracle Rdb Release 7.0.7.1.

5.2.10 GET DIAGNOSTICS Keyword CONNECTION_NAME Returned Incorrect Value

Bug 880810

In previous versions of Oracle Rdb, the GET DIAGNOSTICS keyword CONNECTION_NAME was not always returning the correct value. The following example shows the problem. The result 'RDB$DEFAULT_CONNECTION' was not expected but rather 'MF'.

```
SQL> declare :c char(31);
SQL>
SQL> attach 'alias aa filename sql$database';
SQL> connect to 'alias aa filename db$:mf_personnel' as 'MF';
SQL>
```

```
SQL> set connect 'MF';
SQL> begin
cont> get diagnostic :c = connection_name;
cont> end;
SQL> print :c;
 C
 RDB$DEFAULT_CONNECTION
SQL>
```

This problem has been corrected in Oracle Rdb Release 7.0.7.1. CONNECT now correctly passes the connection name to the Rdb server so that it can be returned by GET DIAGNOSTICS.

5.2.11 Errors Not Reported by SQL

Bug 3083552

In some cases, when a bugcheck was produced, the SQLCODE returned was 0 indicating that no error had occurred.

This problem has been corrected in Oracle Rdb Release 7.0.7.1.

5.2.12 Variable Updated Though No Rows Found

Bug 2863676

Variables in SQL statements were updated even though the statement did not result in any data (for example, no rows selected). This occurred with declared variables in interactive SQL as well as host variables in application programs. The problem is illustrated by the following interactive SQL example.

```
SQL> at 'f personnel';
SQL> declare :my_int integer;
SQL> begin set :my_int = 0; end;
SQL> select 1 from rdb$database where exists
cont> (select * from employees where employee_id = '99999');
0 rows selected
SQL> select 1 into :my_int from rdb$database where exists
cont> (select * from employees where employee_id = '99999');
SQL> print :my_int;
      MY_INT
           1
1 row selected
```

In the above example, the variable ":my_int" should have remained zero since no rows were selected.

As a workaround, the application must examine the SQLSTATE or SQLCODE after each such statement and restore the original value of the variable as needed.

This problem has been corrected in Oracle Rdb Release 7.0.7.1.

5.2.13 Partitioning Clause Not Working in Embedded SQL

Bug 3112810

In some cases, a sequential scan on partitioned data where the partitioning uses a VARCHAR type column can fail (or possibly produce wrong data results).

The following example shows a typical error message:

```
char a_alert[6] = "XYZ05";

EXEC SQL
    SET TRANSACTION READ WRITE WAIT RESERVING
        T_LIMIT PARTITION (3) FOR EXCLUSIVE READ;

EXEC SQL
    SELECT count(*)
    into   :i
    FROM   T_LIMIT
    WHERE  A_ALERT = :a_alert
      and  (
            EXTRACT (YEAR FROM A_DATE_CREATED) * 10000 +
            EXTRACT (MONTH FROM A_DATE_CREATED) * 100 +
            EXTRACT (DAY FROM A_DATE_CREATED)
           ) = 20031017;

%RDB-E-UNRES_REL, relation T_LIMIT in specified request is not a relation
 reserved in specified transaction
-RDMS-E-UNRES_AREA, area 67 within relation "T_LIMIT" is not reserved
```

Rdb is expecting the type of the parameter in the WHERE clause to agree with the type of the column it is comparing against (which is CHAR(5) in this case).

Interactive SQL does the data conversion and comparison correctly but SQL$PRE/CC sometimes gets confused.

Possible workarounds include:

1. Use a compound statement to convert the type from VARCHAR to CHAR.
   Here we recode the example query (above) as a compound statement which assigns the input parameter into an internally declared CHAR(5) variable as follows:

   ```
   EXEC SQL
       BEGIN declare :my_char char(5);
       set :my_char = :a_alert;
       SELECT count(*)
       into   :i
       FROM   T_LIMIT
       WHERE  A_ALERT = :my_char
         and  (
               EXTRACT (YEAR FROM A_DATE_CREATED) * 10000 +
               EXTRACT (MONTH FROM A_DATE_CREATED) * 100 +
               EXTRACT (DAY FROM A_DATE_CREATED)
              ) = 20031017; END;
   ```

The above statement uses the "MY_CHAR" variable to convert the type of the input from VARCHAR (which is the only way that SQL$PRE/CC will send it) into CHAR.

2. Use Dynamic SQL to avoid host variable inputs.
   For example, in the reproducer program declare SQLDA areas as follows:

```
EXEC SQL INCLUDE SQLDA;
...
    struct SQLDA_STRUCT *OUT_AREA;
    OUT_AREA = calloc(1, sizeof(struct SQLDA_STRUCT) +
                      (10*sizeof(struct SQLVAR_STRUCT)) );
    OUT_AREA->SQLN = 10;
```

   Then recode the query as a dynamic statement with no input parameters like this:

```
    sprintf(sql_stmt,
            "SELECT count(*) \
            into   ? \
            FROM   T_LIMIT \
            WHERE  A_ALERT = '%s' \
              and  ( \
                    EXTRACT (YEAR FROM A_DATE_CREATED) * 10000 + \
                    EXTRACT (MONTH FROM A_DATE_CREATED) * 100 + \
                    EXTRACT (DAY FROM A_DATE_CREATED) \
                   ) = 20031017", a_alert);
    EXEC SQL PREPARE stmt1 from :sql_stmt;
    EXEC SQL DESCRIBE stmt1 OUTPUT into :OUT_AREA;
    OUT_AREA->SQLVAR[0].SQLDATA = (char *)&i;
    EXEC SQL EXECUTE stmt1 into descriptor :OUT_AREA;
```

3. Use Dynamic SQL to force host variable parameter inputs to CHAR in an input SQLDA. This method has the advantage that the request can be reused. However, it is imperative that the input strings be space padded to the length of the data (five in our examples).
   For example, declare the SQLDA items as in the Workaround 2 example above plus the following:

```
    struct SQLDA_STRUCT *IN_AREA;
    IN_AREA = calloc(1, sizeof(struct SQLDA_STRUCT) +
                     (10*sizeof(struct SQLVAR_STRUCT)) );
    IN_AREA->SQLN = 10;
```

   Then recode the query to use an input SQLDA as follows:

```
    strcpy(sql_stmt, "SELECT count(*) \
            into   ? \
            FROM   T_LIMIT \
            WHERE  A_ALERT = ? \
              and  ( \
                    EXTRACT (YEAR FROM A_DATE_CREATED) * 10000 + \
                    EXTRACT (MONTH FROM A_DATE_CREATED) * 100 + \
                    EXTRACT (DAY FROM A_DATE_CREATED) \
                   ) = 20031017");

    EXEC SQL PREPARE stmt2 from :sql_stmt;
    EXEC SQL DESCRIBE stmt2 OUTPUT into :OUT_AREA;
    EXEC SQL DESCRIBE stmt2 INPUT into :IN_AREA;
```

```
IN_AREA->SQLVAR[0].SQLTYPE = SQLDA_CHAR;
IN_AREA->SQLVAR[0].SQLDATA = a_alert;
OUT_AREA->SQLVAR[0].SQLDATA = (char *)&i;

EXEC SQL EXECUTE stmt2 into descriptor :OUT_AREA
    using descriptor :IN_AREA;
```

In this example, the second DESCRIBE sets *IN_AREA->SQLVAR[0].SQLTYPE* to *SQLDA_VARCHAR*. But overriding this value to *SQLDA_CHAR* causes dynamic SQL to build a request that works. Again please note that when using *SQLDA_CHAR*, the data string (a_alert in the example) must be space padded to the length specified in *IN_AREA->SQLVAR[0].SQLLEN*.

This problem has been corrected in Oracle Rdb Release 7.0.7.1.

5.3 RDO and RDML Errors Fixed

5.3.1 RDML /DATE_TYPE Qualifier Default is Now NOEMPTY_RECORDS

RDML /PASCAL generates a record type for date items. This type is either an empty record or a record composed of two longword elements depending on the value of the /DATE_TYPE qualifier (EMPTY_RECORDS or NOEMPTY_RECORDS respectively). Earlier versions of the Pascal compiler behaved as desired when empty records were used in assignments and fetches. More current versions of the Pascal compiler give the following warning message when empty records are used for dates:

```
%PASCAL-W-EMPTYVAR, Fetching an empty record with an explicit size
attribute may not yield expected results
```

Furthermore, statements flagged with the above warning frequently do not yield the desired result but instead treat the record as having a zero length. Using the /DATE_TYPE=NOEMPTY_RECORD qualifier avoids this problem.

In order to make RDML /PASCAL more usable, NOEMPTY_RECORDs has been made the default value for the DATE_TYPE. Additionally, using an explicit /DATE_TYPE=EMPTY_RECORDS on the RDML /PASCAL command line will now result in the following warning message being issued by RDML:

```
%RDML-W-DATE_NOEMPTY, /DATE_TYPE=EMPTY_RECORDS specified;
use /DATE_TYPE=NOEMPTY_RECORDS
```

As a workaround, use /DATE_TYPE=NOEMPTY_RECORDS.

This problem has been corrected in Oracle Rdb Release 7.0.7.1.

5.4 Oracle RMU Errors Fixed

5.4.1 RMU /COLLECT May Default to a READ WRITE Transaction

Bug 2898115

When no transaction type is specified for the *RMU/COLLECT OPTIMIZER_STATISTICS* command, RMU examines the database storage areas and if any storage area has snapshots disabled, a read write transaction is used. If no storage areas have snapshots disabled, then a read only transaction is used.

Currently Rdb does not allow different areas on the same database to have snapshots enabled or disabled. However, the attribute is recorded in the database root file against each storage area.

In previous versions, if a database had reserved and unused storage area slots, this check could erroneously examine the unused storage area slots and conclude that areas on the database had snapshots disabled. This could cause RMU to adopt a read write transaction as the default transaction mode for *RMU/COLLECT*.

The following example shows *RMU/COLLECT* executing against a database with reserved storage areas erroneously selecting a read write transaction. The example also shows the use of debug flags to display the transaction modes used by the command.

```
$ DEFINE RDMS$SET_FLAGS TRANSACTION
$ RMU /COLLECT OPTIMIZER MF_PERSONNEL
~T Compile transaction (1) on db: 1
~T Transaction Parameter Block: (len=2)
0000 (00000) TPB$K_VERSION = 1
0001 (00001) TPB$K_WRITE (read write)
~T Start_transaction (1) on db: 1, db count=1
~T Commit_transaction on db: 1
~T Prepare_transaction on db: 1
~T Compile transaction (1) on db: 2
~T Transaction Parameter Block: (len=2)
0000 (00000) TPB$K_VERSION = 1
0001 (00001) TPB$K_READ (read only)
~T Start_transaction (1) on db: 2, db count=1
~T Commit_transaction on db: 2
~T Prepare_transaction on db: 2
~T Compile transaction (1) on db: 3
~T Transaction Parameter Block: (len=2)
0000 (00000) TPB$K_VERSION = 1
0001 (00001) TPB$K_WRITE (read write)
~T Start_transaction (1) on db: 3, db count=1
~T Commit_transaction on db: 3
~T Prepare_transaction on db: 3
~T Compile transaction (1) on db: 4
~T Transaction Parameter Block: (len=2)
0000 (00000) TPB$K_VERSION = 1
0001 (00001) TPB$K_WRITE (read write)
~T Start_transaction (1) on db: 4, db count=1
~T Commit_transaction on db: 4
~T Prepare_transaction on db: 4
```

The following example shows the corrected behaviour.

```
$ RMU /COLLECT OPTIMIZER MF_PERSONNEL /TRANS=READ_ONLY
~T Compile transaction (1) on db: 1
~T Transaction Parameter Block: (len=2)
0000 (00000) TPB$K_VERSION = 1
0001 (00001) TPB$K_READ (read only)
~T Start_transaction (1) on db: 1, db count=1
~T Commit_transaction on db: 1
~T Prepare_transaction on db: 1
~T Compile transaction (1) on db: 2
~T Transaction Parameter Block: (len=2)
0000 (00000) TPB$K_VERSION = 1
0001 (00001) TPB$K_READ (read only)
~T Start_transaction (1) on db: 2, db count=1
~T Commit_transaction on db: 2
~T Prepare_transaction on db: 2
~T Compile transaction (1) on db: 3
~T Transaction Parameter Block: (len=2)
0000 (00000) TPB$K_VERSION = 1
0001 (00001) TPB$K_READ (read only)
~T Start_transaction (1) on db: 3, db count=1
~T Commit_transaction on db: 3
~T Prepare_transaction on db: 3
~T Compile transaction (1) on db: 4
~T Transaction Parameter Block: (len=2)
0000 (00000) TPB$K_VERSION = 1
0001 (00001) TPB$K_WRITE (read write)
~T Start_transaction (1) on db: 4, db count=1
~T Commit_transaction on db: 4
~T Prepare_transaction on db: 4
```

Note that the read write transaction at the end of the example is used to update information in the metadata and is normal and required.

Specifying the transaction type on the command line by using the */TRANSACTION_TYPE=READ_ONLY* qualifier will avoid this problem.

This problem has been corrected in Oracle Rdb Release 7.0.7.1.

5.4.2 RMU/VERIFY/CONSTRAINTS Problems With Named Tables And Constraints

Bug 3016789

Oracle Rdb V7.0 RMU/VERIFY/CONSTRAINTS verified all constraints even if a list of constraints was specified. Oracle Rdb V7.1 RMU/VERIFY/CONSTRAINTS verified all constraints for all tables even if a list of tables was specified. This behaviour has been corrected so that only constraints for a specified list of tables or only the constraints specified will be verified.

The following example shows that for Oracle Rdb V7.0 RMU/VERIFY/CONSTRAINTS, all constraints were verified even if a list of constraints was specified and that for Oracle Rdb V7.1 RMU/VERIFY/CONSTRAINTS, all constraints for all tables were verified even if a list of tables was specified.

```
$DEFINE RDMS$DEBUG_FLAGS "H"
$ rmu/show version
Executing RMU for Oracle Rdb V7.0-7
$ rmu/verify/constraint=(constraint=degrees_foreign2)/log mf_personnel
%RMU-I-BGNROOVER, beginning root verification
%RMU-I-ENDROOVER, completed root verification
%RMU-I-BGNVCONST, beginning verification of constraints for database
device:[directory]MF_PERSONNEL.RDB;1
~H Extension (VERIFY CONSTRAINTS) Item List: (len=0)
~H: ...verify constraint "WORK_STATUS_PRIMARY_STATUS_CODE"
~H: ...verify constraint "STATUS_NAME_VALUES"
~H: ...verify constraint "STATUS_TYPE_VALUES"
~H: ...verify constraint "EMPLOYEES_PRIMARY_EMPLOYEE_ID"
~H: ...verify constraint "EMP_SEX_VALUES"
~H: ...verify constraint "EMP_STATUS_CODE_VALUES"
~H: ...verify constraint "JOBS_PRIMARY_JOB_CODE"
~H: ...verify constraint "WAGE_CLASS_VALUES"
~H: ...verify constraint "DEPARTMENTS_PRIMARY1"
~H: ...verify constraint "JOB_HISTORY_FOREIGN1"
~H: ...verify constraint "JOB_HISTORY_FOREIGN2"
~H: ...verify constraint "JOB_HISTORY_FOREIGN3"
~H: ...verify constraint "SALARY_HISTORY_FOREIGN1"
~H: ...verify constraint "COLLEGES_PRIMARY_COLLEGE_CODE"
~H: ...verify constraint "DEGREES_FOREIGN1"
~H: ...verify constraint "DEGREES_FOREIGN2"
~H: ...verify constraint "DEG_DEGREE_VALUES"
~H: ...verify constraint "CANDIDATES_LAST_NAME_NOT_NULL"
~H: ...verify constraint "RESUMES_UNIQUE_EMPLOYEE_ID"
~H: ...verify constraint "RESUMES_FOREIGN1"
%RMU-I-ENDVCONST, completed verification of constraints for database
device:[directory]MF_PERSONNEL.RDB;1

$ rmu/show version
Executing RMU for Oracle Rdb V7.1-10
$ rmu/verify/constraint=(table=colleges)/log mf_personnel
%RMU-I-BGNROOVER, beginning root verification
%RMU-I-ENDROOVER, completed root verification
%RMU-I-BGNVCONST, beginning verification of constraints for database
device:[directory]MF_PERSONNEL.RDB;1
~H Extension (VERIFY CONSTRAINTS) Item List: (len=0)
~H: ...verify constraint "WORK_STATUS_PRIMARY_STATUS_CODE"
~H: ...verify constraint "STATUS_NAME_VALUES"
~H: ...verify constraint "STATUS_TYPE_VALUES"
~H: ...verify constraint "EMPLOYEES_PRIMARY_EMPLOYEE_ID"
~H: ...verify constraint "EMP_SEX_VALUES"
~H: ...verify constraint "EMP_STATUS_CODE_VALUES"
~H: ...verify constraint "JOBS_PRIMARY_JOB_CODE"
~H: ...verify constraint "WAGE_CLASS_VALUES"
~H: ...verify constraint "DEPARTMENTS_PRIMARY1"
~H: ...verify constraint "JOB_HISTORY_FOREIGN3"
~H: ...verify constraint "JOB_HISTORY_FOREIGN1"
~H: ...verify constraint "JOB_HISTORY_FOREIGN2"
~H: ...verify constraint "COLLEGES_PRIMARY_COLLEGE_CODE"
~H: ...verify constraint "DEGREES_FOREIGN1"
~H: ...verify constraint "DEGREES_FOREIGN2"
~H: ...verify constraint "DEG_DEGREE_VALUES"
~H: ...verify constraint "CANDIDATES_LAST_NAME_NOT_NULL"
~H: ...verify constraint "RESUMES_UNIQUE_EMPLOYEE_ID"
~H: ...verify constraint "RESUMES_FOREIGN1"
%RMU-I-ENDVCONST, completed verification of constraints for database
device:[directory]MF_PERSONNEL.RDB;1
```

This problem has been corrected in Oracle Rdb Release 7.0.7.1.

5.4.3 RMU Unload Incorrectly Using DBKEY SCOPE IS ATTACH

Bug 2623790

In prior releases of Oracle Rdb V7.0, it was possible that RMU Unload would default to DBKEY SCOPE IS ATTACH. This problem has been corrected in Oracle Rdb Release 7.0.7.1.

This problem occurs when applications attach to the Rdb database without a database parameter block (DPB). This can be determined by defining RDMS$SET_FLAGS "DATABASE" prior to running the application or RMU command.

A typical attach to the database will display lines describing the explicitly set "Database Parameter Buffer" including the DBKEY SCOPE.

```
$ SQL$
SQL> attach 'filename SCRATCH';
 ATTACH #1, Database DISK3:[DATABASES.V71]SCRATCH.RDB;1
~P Database Parameter Buffer (version=2, len=78)
0000 (00000) RDB$K_DPB_VERSION2
0001 (00001) RDB$K_FACILITY_ALL
0002 (00002) RDB$K_DPB2_IMAGE_NAME "NODE::DISK1:[DIR]SQL$71.EXE;1"
003F (00063) RDB$K_FACILITY_ALL
0040 (00064) RDB$K_DPB2_DBKEY_SCOPE (Transaction)
0044 (00068) RDB$K_FACILITY_ALL
0045 (00069) RDB$K_DPB2_REQUEST_SCOPE (Attach)
0049 (00073) RDB$K_FACILITY_RDB_VMS
004A (00074) RDB$K_DPB2_CDD_MAINTAINED (No)
 RDMS$BIND_WORK_FILE = "DISK2:[TEST]RDMSTTBL$KMCMLSDFBXK.TMP;" (Visible = 0)
```

In the case of RMU Unload, this buffer is absent and defaults should be used. Unfortunately, the DBKEY SCOPE is undefined and is being incorrectly set to DBKEY SCOPE IS ATTACH.

```
$ RMU /Unload SCRATCH
 ATTACH #1, Database DISK3:[DATABASES.V71]SCRATCH.RDB;1
 RDMS$BIND_WORK_FILE = "DISK2:[TEST]RDMSTTBL$KMY10MNHBXK.TMP;" (Visible = 0)
```

It is possible that other applications or 4GL tools suffer from this problem also. It can be confirmed using RDMS$SET_FLAGS as shown here, or using the RMU/SHOW STATISTICS command to see if any process is waiting for the lock "waiting for database key scope" in the *Active User Stall Messages screen*.

5.5 LogMiner Errors Fixed

5.5.1 LogMiner Elimination of Processing Unneeded 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 ascending 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.

The /RESTART=restart−point qualifier can be used to specify 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.

Previously, all specified input after−image journal files were always processed. This behaviour has been altered when both the RESTART and ORDER_AIJ_FILES qualifiers are specified. In this situation, any after−image journal files containing only sequence numbers prior to the "Micro−quiet point" sequence number (indicated in the AIJ Extract Restart Control Point (AERCP)) can be eliminated from processing after the order of sequence numbers has been determined by the sort operation. Eliminating the unneeded after−image journal files can speed the restart operation.

5.5.2 /TRANSACTION_TYPE Qualifier for RMU /UNLOAD /AFTER_JOURNAL

The RMU /UNLOAD /AFTER_JOURNAL command would start either a read−write or a read−only transaction to read the database metadata. A read−only transaction would be started if the database was set to "SNAPSHOTS ARE IMMEDIATE"; otherwise a read−write transaction would be started.

The qualifier "/TRANSACTION_TYPE=" has been added to the RMU /UNLOAD /AFTER_JOURNAL command to allow explicit control over the transaction type used when reading the database metadata.

The following keywords may be specified to the "/TRANSACTION_TYPE=" qualifier.

- 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.
- READ_ONLY − Starts a READ ONLY transaction.
- WRITE − Starts a READ WRITE transaction.
- ISOLATION_LEVEL − The transaction isolation level. 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.

- ♦ WAIT − Will wait indefinitely on a locked resource.
- ♦ WAIT=n − This instructs Rdb to wait 'n' seconds before aborting the wait and the RMU session. Specifying a wait timeout interval of zero (0) is equivalent to specifying NOWAIT.
- ♦ NOWAIT − Will not wait on locked resources.

If the qualifier /TRANSACTION_TYPE is omitted or specified with no options, then the default is /TRANSACTION_TYPE=(AUTOMATIC, WAIT).

Although a WRITE transaction is started on the database, RMU /UNLOAD /AFTER_JOURNAL does not attempt to write to the database tables.

5.6 RMU Show Statistics Errors Fixed

5.6.1 RMU /SHOW STATISTICS Writes Invalid Configuration File

Bug 3108571

Starting in Oracle Rdb Release 7.0.6.3, the RMU /SHOW STATISICS utility could write an invalid line to a save configuration file. In particular, the "CHECKPOINT_TX" line would be omitted and the line containing "CHECKPOINT_BLOCK_COUNT" would be corrupted.

This problem has been corrected in Oracle Rdb Release 7.0.7.1.

5.7 Hot Standby Errors Fixed

5.7.1 Starting LRS on Master Database Caused Shutdown

Bug 1775983

If a database administrator attempted to start the Hot Standby feature and mistakenly specified the master database as the standby database then the master database was shutdown.

For example, note that in the following command the /MASTER qualifier is being used against the master database. This causes Hot Standby to attempt to configure the master database as a standby database.

```
$ RMU /REPLICATE AFTER_JOURNAL START [.MASTER]mf_personnel −
  /MASTER_ROOT=[.STANDBY]standby_personnel
```

When the above command was issued, the AIJ Log Roll−forward Server (LRS) failed with the following error:

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

Whenever the LRS failed, the database was automatically shutdown. A running application could be accidentally shutdown due to this type of command error.

This problem has been corrected in Oracle Rdb Release 7.0.7.1. Now, if the LRS fails with a STBYDBINUSE error, the database is not shutdown.

Chapter 6
Enhancements Provided in Oracle Rdb Release 7.0.8

6.1 Enhancements Provided in Oracle Rdb Release 7.0.8

6.1.1 Support for OpenVMS Version 8.2

This version of Rdb, Release 7.0.8, supports HP's OpenVMS Version 8.2 Release.

6.1.2 RDM$BIND_SNAP_QUIET_POINT Logical Reinstated

Bug 3908414

Over the years, various problems have been reported related to quiet point backups. In particular, database backups and journal backups would sometimes fail with the following error:

```
%RMU-F-TIMEOUT, timeout on quiet
```

Quiet point backups are required so that recovery of a journal can be done without always requiring previous journals. Full database backups can avoid lock conflicts if they wait for the quiet point lock. See the documentation for the RMU Backup commands for more information regarding quiet point backups.

Lock timeout errors for the quiet lock are intended to be returned when a long running update transaction has not completed within the timeout period. However, Oracle Rdb Release 6.0 forced all transactions (read−only or read−write) to obtain the quiet point lock when starting. That greatly increased the incidence of timeout errors from backups. To remedy this situation, a special logical name, RDM$BIND_SNAP_QUIET_POINT, was implemented that would force processes starting read−only transactions to release the quiet point lock. If that logical was defined to the value 0 then read−only transactions would not obtain the quiet point lock.

Defining the RDM$BIND_SNAP_QUIET_POINT logical to 0 would usually resolve problems with timeouts on the quiet lock, but it would effectively disable the fast commit performance feature for applications that often switched between read−only and read−write transactions. (See Bug 884004.) To remedy that situation, the transaction start code was modified to retain the quiet lock during read−only transactions but release the lock during the read−only transaction if a backup process started. With this change, the RDM$BIND_SNAP_QUIET_POINT lock logical could be defined without impacting the performance of the fast commit feature. That change was introduced in Releases 7.0.6.3 and 7.1.0.1.

The previous fix resolved performance problems, but the logical still could not be defined to 0 if the Hot Standby feature was being used. (See Bug 2656534.) If the Hot Standby feature was being utilized, the logical had to be undefined, or defined to the default value of 1. Applications that utilized Hot Standby were still subject to undeserved timeouts from backup commands. In Releases 7.0.7.1 and 7.1.2, the Hot Standby feature was modified so that read−only transactions were not required to hold the quiet point lock. Also, because read−only processes would usually release the quiet point lock when a backup started, the RDM$BIND_SNAP_QUIET_POINT logical was completely removed.

After the above changes, quiet lock timeouts were no longer an issue for most applications. However, a read−only transaction would only release the quiet point lock when Oracle Rdb had returned control to the user application. Some complicated queries could execute for an exceptionally long period of time before returning a row to the user application and thus might not release the quiet lock in time to prevent timeout errors for the backup process. For example, read−only queries that executed aggregate functions such as SUM or AVERAGE on large sets of rows, or queries that required sorts of large sets of rows before any rows were returned could sometimes not release the quiet point lock before the backup command timed out. (See Bug 3908414.) In addition, any update transaction that started after the backup process requested the quiet point lock would stall until the long running read−only request returned control to the user application. That means that it was possible for many database users to stall waiting for the read−only transaction to complete, even though they had released the quiet point lock.

This release of Oracle Rdb reinstates the RDM$BIND_SNAP_QUIET_POINT logical so that read−only transactions can be forced to release the quiet point lock before starting. The logical now has a slightly different meaning than the original implementation. The default value is still 1, but that value now signifies that all transactions will hold the quiet point lock until a backup process requests it. Read−only transactions will not obtain the quiet point lock; only read−write requests will obtain the quiet point lock. This is the behavior that was introduced in response to Bug 884004. If the logical is defined to be 0, then read−only transactions will always release the quiet point lock at the beginning of the transaction, regardless of the existence of a backup process. That implies that all modified buffers in the buffer pool have to be written to disk before the transaction proceeds. Applications that utilize the fast commit feature and often switch between read−only and read−write transactions within a single attach may experience performance degradation if the logical is defined to 0. This is the behavior that was in place prior to Releases 7.0.6.3 and 7.1.0.1.

Oracle Corporation recommends that you do not define the RDM$BIND_SNAP_QUIET_POINT logical for most applications. If the scenario described in Bug 3908414 is being encountered, the logical can be defined to 0 to force read−only transactions to always release the quiet point lock. Rather than define the logical system−wide, this logical can be defined for specific jobs that are likely to execute for an extensive period of time before returning to the user application. This parameter may also be manipulated from the RMU Show Statistics locking dashboard. That way most users will not have to sacrifice fast commit performance when switching between read−only and read−write transactions. As of releases 7.0.7.1 and 7.1.2, Hot Standby is no longer affected by this logical so Hot Standby is not a factor when determining how to define the logical.

6.1.3 RMU Unload After_Journal/Ignore Old_Version Keyword

The RMU Unload After_Journal command treats non−current record versions in the AIJ file as a fatal error condition. That is, attempting to extract a record that has a record version not the same as the table's current maximum version results in a fatal error.

There are, however, some very rare cases where a verb rollback of a modification of a record may result in an old version of a record being written to the after−image journal even though the transaction did not actually complete a successful modification to the record. The RMU Unload After_Journal command detects the old record version and aborts with a fatal error in this unlikely case.

The RMU Unload After_Journal command now accepts a new keyword to partially work around this case. The Ignore qualifier has been enhanced to include the keyword Old_Version. When this keyword is present, the RMU Unload After_Journal command displays a warning message for each record that has a non−current record version and the record is not written to the output stream. The Old_Version keyword accepts an optional list of table names to indicate that only the specified tables are permitted to have non−current record version errors ignored.

6.1.4 New Features in RMU Extract

There have been several enhancements to the RMU Extract command. The following description of the command is the RMU Extract command chapter as it would appear in the *Oracle RMU Reference Manual* for Release 7.0.8.

RMU Extract Command

Reads and decodes Oracle Rdb metadata and reconstructs equivalent statements in Relational Database Operator (RDO) or SQL (structured query language) code for the definition of that database. These statements can either be displayed or extracted. You can use these statements to create your database again if you no longer have the RDO or SQL code that defined your database.

In addition, you can direct the RMU Extract command to produce output for the following:

- An SQL or RDO IMPORT script (Items=Import)
- An RMU Unload command for each table (Items=Unload)
- An RMU Load command for each table (Items=Load)
- An RMU Set Audit command for the database (Items=Security)
- An RMU Verify command for the database (Items=Verify)

RMU/Extract root-file-spec

Command Qualifiers	Defaults
/Items[=item-list]	/Items=All
/Language=lang-name	/Language=SQL
/[No]Log[=log-file]	/Nolog
/Options=options-list	/Option=Normal
/[No]Output[=out-file]	/Output=SYS$OUTPUT
/Transaction_Type[=(transation_mode,options...)]	See Description

---

DESCRIPTION

The RMU Extract command decodes information and reconstructs equivalent commands in the language you select with the Language qualifier for the definition of that database.

You can extract the definitions to either a file or to SYS$OUTPUT. The RMU Extract command extracts the following character set information:

- For databases:
  - The database default character set
  - The national character set
- For domains:
  - The character set of each character data type domain
  - The length in characters of each character data type domain
- For tables:
  - The character set of each character data type column
  - The length in characters of each character data type column

The RMU Extract command may enclose object names in double quotation marks to preserve the uppercase and lowercase characters, to represent different character sets, or to allow processing of SQL reserved keywords.

COMMAND PARAMETERS

root−file−spec

The file specification for the database root file from which you want to extract definitions. Note that you do not need to specify the file extension. If the database root file is not found, the command exits with a "file not found" error.

COMMAND QUALIFIERS

Items[=item−list]

Allows you to extract and display selected definitions. Note that each of the item names can be combined to provide shorter command lines such as the following:

```
$ RMU/EXTRACT/NOLOG/ITEMS=(ALL,NODATABASE) MF_PERSONNEL
```

If you omit the Items qualifier from the command line or specify it without any options, the action defaults to Items=All.

The following options can be specified with the Items qualifier:

- All
  Indicates that all database items are to be extracted. This is the default and includes all items except Alter_Database, Revoke_entry, Import, Load, Protections, Security, Unload, Verify, Volume, and Workload options. You can use either All or Noall in combination with other items to select specific output.
  In the following example, the Items=All option causes all the definitions except for Triggers to be extracted and displayed:

  ```
  $ RMU/EXTRACT/ITEMS=(ALL,NOTRIGGERS) MF_PERSONNEL
  ```

The following example displays domain and table definitions. Note that the Noall option could have been omitted:

```
$ RMU/EXTRACT/ITEMS=(NOALL, DOMAIN, TABLE) MF_PERSONNEL
```

◊ Alter_Database (or Change_Database)
Displays the physical database after−image journal object definition.

◊ Catalog
Displays all contents of the catalog created for an SQL multischema database. This item is ignored if the interface is RDO.

◊ Collating_Sequences
Displays all the collating sequences defined for the database that you select. Note that Oracle Rdb does not save the name of the source OpenVMS National Character Set (NCS) library and the name becomes the defined logical, NCS$LIBRARY, by default.

◊ Constraints
Noconstraints
Table and column constraints are included in the output of the Items=Table qualifier. If you specify Item=Noconstraints, constraint information is not extracted for any table. If you specify the Language=SQL qualifier, the default is to have Item=Constraints enabled when tables are extracted.
To extract all constraints as an ALTER TABLE statement, use the Item=Constraint and Option=Defer_Constraints qualifiers. To force all constraints to be defined after tables are defined, use the Item=Tables and Option=Defer_Constraints qualifiers.

◊ Database
Displays the database attributes and characteristics. This includes information such as the database root file name, the number of buffers, the number of users, the repository path name, and the characteristics for each storage area. If you specify RMU Extract with the Option=Nodictionary_References qualifier, the data dictionary path name is ignored.

◊ Domains (or Fields)
Displays the domain definitions. If the domain was originally defined using the data dictionary path name, the output definition shows this. If the Option=Nodictionary_References qualifier is specified, the data dictionary path name is ignored and the column attributes are extracted from the system tables.

◊ Functions
Displays external function definitions.

◊ Import
Generates an RDO or SQL IMPORT script that defines every storage area and row cache. The Language qualifier determines whether Oracle RMU generates an RDO or SQL IMPORT script (If you specify the Language=SQL or the Language=ANSI_SQL qualifier, the same SQL IMPORT script is generated.) Because the RDO interface does not accept many of the database options added to recent versions of Oracle Rdb, Oracle Corporation recommends that you specify the Language=SQL qualifier (or accept the default).
The Items=Import qualifier is useful when you want to re−create a database that is the same or similar to an existing database. Editing the file generated by Oracle RMU to change allocation parameters or add storage areas and so

on is easier than writing your own IMPORT script from scratch.
When Oracle RMU generates the IMPORT script, it uses an interchange file name of rmuextract_rbr in the script. Therefore, you must either edit the IMPORT script generated by Oracle RMU to specify the interchange file that you want to import, or assign the logical name RMUEXTRACT_RBR to your interchange file name. (An interchange file is created by an SQL or RDO EXPORT statement.) See Example 14 in the Examples section.

◊ Indexes (or Indices)
Displays index definitions, including storage map information.

◊ Load
Unload
Generates a DCL command procedure containing an RMU Load or RMU Unload command for each table in the database. This item must be specified explicitly, and is not included by default when you use the Items=All qualifier.
Oracle RMU generates the Fields qualifier for the Load and Unload scripts when you specify the Option=Full qualifier. If you do not specify the Option=Full qualifier, the scripts are generated without the Fields qualifier.
If you specify the RMU Extract command with the Item=Unload qualifier, DCL commands are added to the script to create a file with type .COLUMNS. This file defines all the unloaded columns. The file name of the .COLUMNS file is derived from the name of the extracted table. You can reference the file by using the "@" operator within the Fields qualifier for the RMU Load and RMU Unload commands.

◊ Module
Displays procedure and function definitions. This item is valid only when the Language specification is SQL; it is ignored if the Language specification is RDO or ANSI_SQL.

◊ Outlines
Displays query outline definitions. This item is valid only when the Language specification is SQL; it is ignored if the Language specification is RDO or ANSI_SQL.

◊ Procedures
Extracts external procedures.

◊ Protections
Displays the protection access control list (ACL) definitions. If the protections are defined using SQL ANSI semantics, they cannot be represented in RDO. In this case, the diagnostic message warns you that the protections must be extracted using the Language=SQL qualifier. If you specify Language=ANSI_SQL, a diagnostic message warns you that the ACL−style protections cannot be extracted in ANSI format. You must explicitly specify the Protections option. It is not included by default when you use the Items=All qualifier.

◊ Revoke_Entry
Extracts a SQL or RDO script that deletes the protections from all access control lists in the database: database, table, column, module, function, and procedure.
The output script contains a series of SQL REVOKE ENTRY statements (or DELETE PROTECTION statements if the language selected is RDO) that remove the access control entry for the user and all objects.

◊ Schema
Displays the schema definitions for an SQL multischema database. This option is ignored if the interface is RDO.

◊ Security
Displays RMU Audit commands based on information in the database. This item must be specified explicitly, and is not included by default when you use the Items=All qualifier.

◊ Storage_Maps
Displays storage map definitions, including the list (segmented string) storage map.

◊ Tables (or Relations)
Displays table definitions in the same order in which they were created in the database.
If the table was originally defined using the data dictionary path name, that path name is used for the definition.
If you specify the Option=Nodictionary_References qualifier, the data dictionary path name is ignored and the table attributes are extracted from the system tables.
If Item=Noconstraints is specified, constraint information is not extracted for any table.
The Items=Tables qualifier handles domains in the following ways:

- The output for this item reflects the original definitions. If a column is based on a domain of a different name, the BASED ON clause is used in RDO, and the domain name is referenced by SQL.
- Any columns that are based on fields in a system table are processed but generate warning messages.
- Certain domains created using RDO in a relation definition cannot be extracted for RDO because it is not possible to distinguish columns defined using a shorthand method as shown in the example that follows. In this case, the column FIELD_1 becomes or is defined as a domain.

```
DEFINE RELATION REL1.
    FIELD_1     DATATYPE IS TEXT SIZE 10.
END.
```

However, this type of definition in SQL causes special domains to be created with names starting with SQL$. In this case, the SQL domain is translated into the following data type:

```
CREATE TABLE TAB1
   (COLUMN_1   CHAR(10));
```

The output for this item also includes the table−level constraints that can be applied: PRIMARY KEY, FOREIGN KEY, NOT NULL, UNIQUE, and CHECK. In the case of the CHECK constraint, the expression might not be translated to or from RDO and SQL due to interface differences.

◊ Triggers
Displays trigger definitions.

◊ Verify
Causes the generation of an optimal DCL command procedure containing

multiple RMU Verify commands. Using this command procedure is equivalent to performing a full verification (RMU Verify with the All qualifier) for the database. This command procedure can be broken down further into partial command scripts to perform partial verify operations. These partial command scripts can then be submitted to different batch queues to do a full verify operation in parallel, or they can be used to spread out a full verify operation over several days by verifying a piece of the database at a time.

A ***partitioning algorithm*** is a procedure to determine what portions of the database should be verified in the same command script. For example, areas with interrelations should be verified with the same partial command script. A partitioning algorithm considers the following when creating a partial command script from the equivalent RMU Verify command with the All qualifier:

1. Each storage area is assigned to a partition.
2. For each table in the database, if the table is not partitioned, the table is put in the partial command script corresponding to that storage area; otherwise, if the table is partitioned across several storage areas, the partitions corresponding to all of the storage areas are merged into one partial command script and the table is added to this partial command script.
3. For each index in the database, the process shown in step 2 is followed.
4. For an index on a table, the index and table are merged into one partial command script.

The scripts of partial RMU Verify commands are written in the form of a command procedure. Each partial command script is preceded by a label of the form STREAM_n: where *n* is an integer greater than 1. For example, to execute the command at label STREAM_3:, invoke the command procedure by using the following syntax:

```
$ @<command-procedure-name> STREAM_3
```

The resultant command procedure is set up to accept up to four parameters, P1, P2, P3, and P4, as shown in Table 6−1.

Table 6−1 Parameters for Generated Command File

Parameter	Option	Description
P1	Stream_n	Specifies the command stream to be executed. The variable *n* is the "number" of the RMU Verify command stream to be executed. If omitted, all command streams are executed.
P2	[No]Log	Specifies whether to use the Log qualifier in the RMU Verify command line. If omitted, the DCL verify switch value is used.
P3		

	Read_Only | Protected | Exclusive	Provides the RMU Verify Transaction_Type value. If omitted, Transaction_Type = Protected is used.
P4		Specifies the name of the output file for the RMU Verify Output qualifier. If omitted, Output = SYS$OUTPUT is used.

◊ Views

Displays view definitions. If the database was defined using SQL, it is possible that the view cannot be represented in RDO. In this case, the diagnostic message warns that the view definition is being ignored, and the user should use LANGUAGE=SQL to extract the view. Note the following transformations the RMU Extract command makes when it cannot precisely replicate the SQL source code:

· The RMU Extract command cannot precisely replicate derived table column names or correlation names for any select expression.
The RMU Extract command generates new names for correlation names (C followed by a number) and derived table column names (F followed by a number). In some cases the derived table column names will be inherited from the nested table columns.
For example, suppose you create a view, as follows:

```
SQL> ATTACH 'FILENAME mf_personnel';
SQL> CREATE VIEW DERIVED_1
cont> (F1) AS
cont> SELECT CAST(AVG(JOB_COUNT) AS INTEGER(2))
cont> FROM (SELECT EMPLOYEE_ID, COUNT (*)
cont>   FROM JOB_HISTORY
cont>   GROUP BY EMPLOYEE_ID) AS EMP_JOBS (EMPLOYEE_ID, JOB_COUNT);
SQL> COMMIT;
```

If you issue the following RMU Extract command, you receive the output shown:

```
$ rmu/extract/item=view/option=(match:DERIVED_1%,noheader,filename_or
mf_personnel
set verify;
set language ENGLISH;
set default date format 'SQL92';
set quoting rules 'SQL92';
set date format DATE 001, TIME 001;
attach 'filename MF_PERSONNEL';
create view DERIVED_1
    (F1) as
    (select
        CAST(avg(C2.F2) AS INTEGER(2))
    from
        (select C4.EMPLOYEE_ID, count(*)
            from JOB_HISTORY C4
            group by C4.EMPLOYEE_ID)
        as C2 (F1, F2));

commit work;
```

· The RMU Extract command cannot generate the original SQL source code for the user−supplied names of AS clauses. This is particularly

apparent when the renamed select expression is referenced in an ORDER BY clause. In such a case, the RMU Extract command generates correlation names in the form RMU$EXT_n where *n* is a number.
For example, suppose you create a view, as follows:

```
SQL> SET QUOTING RULES 'SQL92';
SQL> CREATE DATA FILE xyz;
SQL> CREATE TABLE DOCUMENT
cont> (REPORT CHAR(10));
SQL> CREATE TABLE REPORTING
cont> (NAME CHAR(5));
SQL> CREATE TABLE "TABLES"
cont> (CODTAB CHAR(5));
SQL> CREATE VIEW VIEW_TEST
cont> (CREDIT,
cont> CODTAB,
cont> CODMON) AS
cont> SELECT
cont> C1.NAME,
cont> C2.CODTAB,
cont> (SELECT C7.REPORT FROM DOCUMENT C7) AS COM
cont> FROM REPORTING C1, "TABLES" C2
cont> ORDER BY C1.NAME ASC, C2.CODTAB ASC, COM ASC;
SQL>
```

If you issue the following RMU Extract command, you receive the output shown:

```
$ RMU/EXTRACT/ITEM=VIEW XYZ.RDB

   .
   .
   .
create view VIEW_TEST
    (CREDIT,
     CODTAB,
     CODMON) as
    select
        C1.NAME,
        C2.CODTAB,
        (select DOCUMENT.REPORT from DOCUMENT) AS RMU$EXT_1
    from REPORTING C1, "TABLES" C2
    order by C1."NAME" asc, C2.CODTAB asc, RMU$EXT_1 asc;
```

◊ Volume
Displays cardinality information in a PDL−formatted file for use by Oracle Expert for Rdb. This item must be specified explicitly, and is not included by default when the Items=All qualifier is used.

◊ Workload
Generates a DCL command language script. The script is used with the RMU Insert Optimizer_Statistics command to extract the work load and statistics stored in the RDB$WORKLOAD table. The unloaded information can be applied after a new database is created using the SQL EXPORT and IMPORT statements, or it can be applied to a similar database for use by the RMU Collect Optimizer_Statistics/Statistic=Workload command.

This item must be specified explicitly, and is not included by default when the Items=All qualifier is used. The default is Noworkload.
You can modify the output of the Item=Workload qualifier by specifying the following keywords with the Option qualifier:

- Audit_Comment
  Each RMU Insert Optimizer_Statistics statement is preceded by the created and altered date for the workload entry. The default is Noaudit_comment.
- Filename_Only
  The database file specification output for the RMU Insert Optimizer_Statistics statement is abbreviated to just the filename.
- Match
  A subset of the workload entries based on the wildcard file name is selected.

Language=lang−name

Allows you to select one of the following interfaces:

◊ SQL
When you specify the Language=SQL qualifier, Oracle RMU generates the Oracle Rdb SQL dialect. The Oracle Rdb SQL dialect is a superset of SQL92 Entry level, with language elements from Intermediate and Full SQL92 levels. It also contains language elements from SQL:1999 and extensions specific to Oracle Rdb.

◊ ANSI_SQL
When you specify the Language=ANSI_SQL qualifier and specify the Option=Normal qualifier, Oracle RMU tries to generate ANSI SQL statements that conform to the ANSI X3.135−1989 SQL standard.
When you specify the Language=ANSI_SQL qualifier and the Option=Full qualifier, Oracle RMU tries to generate SQL statements that conform to the current ANSI and ISO SQL Database Language standards. Please refer to the Oracle Rdb SQL Reference Manual for details on the conforming SQL language standards.
Regardless of the Option parameter you specify, any Oracle Rdb specific features (such as DATATRIEVE support clauses and storage maps) are omitted.

◊ RDO
When you specify the RDO language option, Oracle RMU generates RDO statements.

The default is Language=SQL.

The Language qualifier has no effect on the output generated by the Items=Load, Items=Unload, and Items=Verify qualifiers. This is because these qualifiers generate scripts that contain Oracle RMU commands only.

Log[=log−file]

Nolog

Enable or disables log output during execution of the RMU Extract command. The log includes the current version number of Oracle Rdb, and the values of the parameter and qualifiers. The default is Nolog. The default file extension is .log. If you specify Log without specifying a file name, output is sent to SYS$OUTPUT.

Options=options−list

This qualifier is used to change the output of the RMU Extract command. This qualifier is not applied to output created by the Items=Unload, Items=Load, Items=Security, or the Items=Verify qualifier.

The following options can be specified with the Options qualifier:

- Audit_Comment
  Noaudit_Comment
  Annotates the extracted objects with the creation and last altered timestamps as well as the username of the creator. The date and time values are displayed using the current settings of SYS$LANGUAGE and LIB$DT_FORMAT. Noaudit_Comment is the default.
- Cdd_Constraints
  Nocdd_Constraints
  Specifies that tables extracted by pathname include all constraints. The Option=Nocdd_Constraints qualifier is equivalent to the Option=Defer_Constraints qualifier for tables with a pathname. This option is ignored if Item=Noconstraints is specified.
  When you specify the Cdd_Constraints option and the Dictionary_References option, the RMU Extract command does not generate ALTER TABLE statements to add constraints, but instead assumes they will be inherited from the data dictionary.
  When you use the Nocdd_Constraints option and the Dictionary_References option, the RMU Extract command generates ALTER TABLE statements to add FOREIGN KEY and CHECK constraints after all base tables have been created.
- Cdd_References
  This option is an alias for Dictionary_References.
- Column_Volume
  Directs the RMU Extract command to output the table, column, and column segmented string cardinalities based on sorted indexes. Note that this qualifier must be used in combination with the Items=Volume qualifier. If the Items=Volume qualifier is omitted, cardinalities are not displayed.
  RMU Extract generates data of the following type:

```
Volume for schema MF_PERSONNEL
    Default volatility is 5;
    Table WORK_STATUS all is 3;
    Table EMPLOYEES all is 100;
        Column EMPLOYEE_ID all is 100;
        Column LAST_NAME all is 83;
   .
   .
```

```
.
 Table RESUMES all is 3;
     List RESUME
         Cardinality IS 3
             Number of segments is 3
             Average length of segments is 24;
```

◊ Debug
Dumps the internal representation for SQL clauses such as VALID IF, COMPUTED BY, MISSING_VALUE, DEFAULT_VALUE, CONSTRAINTS, SQL DEFAULT, TRIGGERS, VIEWS, and STORAGE MAPS during processing. The keyword Debug cannot be specified with the keywords Normal or Full in the same Options qualifier list.

◊ Defer_Constraints
Nodefer_Constraints
Forces all constraints to be defined (using an ALTER TABLE statement) after all tables have been extracted. This option is ignored if Item=Noconstraints is specified.
If Option=Nodefer_Constraints is specified, all constraints are generated with the CREATE TABLE statement. If neither Option=Defer_Constraints nor Option=Nodefer_Constraints is specified, the default behavior is to generate NOT NULL, UNIQUE, and PRIMARY KEY constraints with the CREATE TABLE statement, and generate CHECK and FOREIGN KEY constraints in a subsequent ALTER TABLE statement.

◊ Dictionary_References
Nodictionary_References
Directs the RMU Extract command to output definitions for domains (fields) and tables (relations) that reference data dictionary path names rather than using the information contained in the Oracle Rdb system tables. In addition to the database statements, this option also displays the data dictionary path name stored in the database. Refer to Example 8 in the Examples section for an example of using this option.
If neither the Option=Dictionary_References qualifier nor the Option=Nodictionary_References qualifier is specified, then Oracle RMU examines the RDB$RELATIONS and RDB$FIELDS system tables to determine whether or not any domains or tables refer to the data dictionary. If references are made to the data dictionary, then the Option=Dictionary_References qualifier is the default. Otherwise, it is assumed that the data dictionary is not used, and the default is the Option=Nodictionary_References qualifier.
The Nodictionary_References keyword causes all references to the data dictionary to be omitted from the output. This is desirable if the database definition is to be used on a system without the data dictionary or in a testing environment.
If the Items=Database and Option=Nodictionary_References qualifiers are selected, the data dictionary path name stored in the system table is ignored. For SQL, the no PATHNAME clause is generated, and for RDO, the clause DICTIONARY IS NOT USED is generated.
If the Items qualifier specifies Domain or Table, and the Option qualifier specifies Nodictionary_References, the output definition includes all attributes stored in the system tables.

◊ Disable_Objects
Nodisable_Objects

Requests that all disabled objects be written to the RMU Extract output file as disabled (see the description for the Omit_Disabled option).
Disable_Objects is the default.
The Nodisable_Objects option displays the objects but omits the disabling syntax.

◊ Domains
Nodomains
The Nodomains option is used to eliminate the domain name from within metadata objects. The domain name is replaced by the underlying data type. This option is designed for use with tools that do not recognize this SQL92 SQL language feature.
Effect on /Language=SQL output:

· The default is Option=Domains.
A SQL script generated when Option=Nodomains was specified does not include the domain name in the CREATE TABLE column definition, CREATE FUNCTION or CREATE PROCEDURE parameter definitions, or any value expression which uses the CAST function to convert an expression to a domain data type (such as the CREATE VIEW and CREATE TRIGGER statements).
The output generated by RMU Extract for functions and procedures in the CREATE MODULE statement is not affected by the Option=Nodomains option because it is based on the original source SQL for the routine body which is not edited by the RMU Extract command.

Effect on /Language=ANSI_SQL output:

· The default is Option=Nodomains when the Option=Normal qualifier is specified or is the default. The RMU Extract command does not generate a list of domain definitions even if the Items=Domains or Items=All qualifier is used. If you want the generated script to include a list of domain definitions, use the Options=Domains qualifier:

```
$RMU/EXTRACT/LANGUAGE=ANSI_SQL/OPTION=DOMAINS databasename
```

Use the Option=Full qualifier to have the use of domains included in the syntax generated for SQL92.

◊ Filename_Only
Causes all file specifications extracted from the database to be truncated to only the file name. The use of this qualifier allows for easier relocation of the new database when you execute the created procedure.

◊ Full
Nofull
Specifies that if metadata that cannot be translated from the language that defined the database to the equivalent construct in the language specified with the Language qualifier (for example, DEFAULT for SQL and the language selected was RDO) then the metadata is displayed in comments, or Oracle RMU attempts to create a translation that most closely approximates the original construct.
Nofull is identical to the Normal option.

◊ Group_Table
Nogroup_Table
Specifies that indexes and storage maps are to be extracted and grouped by table. The table is extracted first, than any PLACEMENT VIA index, then any storage map, and finally all other indexes.
When the Group_Table qualifier is combined with the Option=Match qualifier, you can select a table and its related storage map and indexes.
The default behavior is Nogroup_Table, which means that items are extracted and grouped by type.

◊ Header
Noheader
Specifies that the script header (and section headers) are included in the extract. This is the default. Noheader suppresses the header and because of the included date may allow easier comparison with other database extractions using the OpenVMS DIFFERENCES command.

◊ Limit_Volume=nn
Nolimit_Volume
Specifies the maximum amount of data to be scanned for segmented fields. The RMU Extract command stops scanning when the limit *nn* is reached. The number of segments and average length of segments are calculated from the data that was scanned. Limit_Volume=1000 is the default.
Nolimit_Volume specifies that a full scan for segmented strings should be done.

◊ Match:match−string
The Match option allows selection of wildcard object names from the database. The match string can contain the standard SQL wildcard characters: the percent sign (%) to match any number of characters, and the underscore (_) to match a single character. In addition, the backslash (\) can be used to prefix these wildcards to prevent them from being used in matching. If you are matching a literal backslash, use the backslash twice, as shown in the following example:

```
Option=Match:"A1\\A2%"
```

The match string defaults to the percent sign (%) so that all objects are selected. To select those objects that start with JOB, use the qualifier Option=Match:"JOB%".
From the mf_personnel database, this command displays the definitions for the domains JOB_CODE_DOM and JOB_TITLE_DOM, the tables JOBS and JOB_HISTORY, the index JOB_HISTORY_HASH, and the storage maps JOBS_MAP and JOB_HISTORY_MAP.
The match string can be quoted as shown if the string contains spaces or other punctuation characters used by DCL or other command language interfaces. Most object names are space filled; therefore, follow the match string with the percent sign (%) to match all trailing spaces.
The Match option can be used in conjunction with the Item qualifier to extract specific tables, indexes, and so on, based on their name and type.
If Group_Table is specified, the match name is assumed to match a table name; indexes for that table will be extracted when the Items=Index qualifier is specified.

◊ Multischema
Nomultischema
Displays the SQL multischema names of database objects. It is ignored by the Relational Database Operator (RDO).
The Nomultischema option displays only the SQL single−schema names of database objects.

◊ Normal
Includes only the specific source language code used to define the database. This is the default.
In addition, this option propagates RDO VALID IF clauses as column CHECK constraints with the attribute NOT DEFERRABLE when the Language specification is SQL or ANSI_SQL. When an RDO VALID IF clause is converted, Oracle RMU generates error messages similar to the following in your log file:

```
%RMU−W−UNSVALIDIF, VALID IF clause not supported in SQL − ignored
 for DEGREE.
%RMU−I−COLVALIDIF, changed VALID IF clause on domain DEGREE to
 column check constraint for DEGREES.DEGREE
```

The first message is a warning that the VALID IF clause could not be added to the domain definition because the VALID IF clause is not supported by SQL. The second message is an informational message that tells you the VALID IF clause was changed to a column check constraint.

◊ Omit_Disabled
Noomit_Disabled
Causes all disabled objects to be omitted from the output of the RMU Extract command. This includes indexes that have MAINTENANCE IS DISABLED, USERS with ACCOUNT LOCK, and disabled triggers and constraints.
The Noomit_Disabled option causes all disabled objects to be included in the output from the RMU Extract command. Noomit_Disabled is the default.

◊ Order_By_Name
Noorder_By_Name
Order_by_Name displays the storage area, cache, and journal names for the items Database, Alter_Database (also known as Change_Database), and Import in alphabetic order by the ASCII collating sequence.
Noorder_By_Name displays the storage area, cache, and journal names for the items Database, Alter_Database, and Import in approximate definition order. The default ordering is approximate because a DROP STORAGE AREA, DROP CACHE, or DROP JOURNAL statement frees a slot that can be reused, thus changing the order. Noorder_By_Name is the default.
You can use the logical name RDMS$BIND_SORT_WORKFILES to allocate work files, if needed.

Note

If the identifier character set for the database is not MCS or ASCII, then this option is ignored. Characters from other character sets do not sort appropriately under the ASCII collating sequence.

◊ Volume_Scan
Directs the RMU Extract command to perform queries to calculate the cardinality of each table, if both the Items=Volume and Options=Volume_Scan qualifiers are specified. The default is Options=Novolume_Scan, in which case the approximate cardinalities are read from the RDB$RELATIONS system table. The Options=Volume_Scan option is ignored if the Items=Volume qualifier is not selected.

◊ Width=n
Specifies the width of the output files. You can select values from 60 to 512 characters. The default of 80 characters is appropriate for most applications.

Output=[out−file]

Nooutput

Names the file to which the RMU Extract command writes the data definition language (DDL) statements. The file extension defaults to .rdo, if you specify the Language=RDO qualifier; .sql, if you specify either the Language=SQL or the Language=ANSI_SQL qualifier. If you specify the Volume option only, the output file type defaults to .pdl. If you specify Load, Security, Verify, or Unload only, the output file type defaults to .com. The default is SYS$OUTPUT. If you disable the output by using the Nooutput qualifier, command scripts are not written to an output file. The Log output can be used to determine which features used by the database cannot be converted to SQL.

Table 6−2 shows the effects of the various combinations of the Language and Options qualifiers.

Table 6−2 Using Qualifiers to Determine Output Selection

Language	Option	Effect on Output
RDO	Normal	Generates RDO syntax.
	Full	Generates RDO syntax.
	Dictionary_ References	Outputs path name references to the repository.
	Nodictionary_ References	Converts path name references to the repository to RDO syntax.
	Multischema	Ignored by RDO.
SQL	Normal	Generates SQL syntax.
	Full	Tries to convert RDO specific features to SQL (for example, the VALID IF clause).
	Dictionary_ References	Outputs path name references to the data dictionary.
	Nodictionary_ References	Converts path name references to the data dictionary to SQL syntax.
	Multischema	Selects SQL multischema naming of objects.
ANSI_SQL	Normal	Generates ANSI/ISO syntax.

	Full	Generates ANSI/ISO SQL92 syntax supported by SQL.
	Dictionary_ References	Ignored for ANSI_SQL.
	Nodictionary_ References	Converts path name references to the data dictionary to SQL syntax. This is the default for ANSI_SQL.
	Multischema	Selects SQL multischema naming of objects.
Any	Audit_Comment	Adds a comment before each definition.
	Debug	Annotates output where possible.
	Domains	Replaces domain names for CAST expression, column and parameter definitions, and returns clauses with SQL data type.
	Filename_Only	Truncates all file specifications extracted from the database to only the file name.
	Volume_Scan	Forces a true count of Tables. Only valid for Items=Volume.

Transaction_Type[=(transaction_mode,options,...)]

Allows you to specify the transaction mode, isolation level, and wait behavior for transactions.

Use one of the following keywords to control the transaction mode:

- ◊ Automatic
  When Transaction_Type=Automatic is specified, the transaction type depends on the current database settings for snapshots (enabled, deferred, or disabled), transaction modes available to the process, and the standby status of the database. Automatic mode is the default.
- ◊ Read_Only
  Starts a READ ONLY transaction.
- ◊ Write
  Starts a READ WRITE transaction.

Use one of the following options with the keyword Isolation_Level=[level] to specify the transaction isolation level:

- ◊ Read_Committed
- ◊ Repeatable_Read
- ◊ Serializable. Serializable is the default setting.

Refer to the SET TRANSACTION statement in the Oracle Rdb SQL Reference Manual for a complete description of the transaction isolation levels.

Specify the wait setting by using one of the following keywords:

- ◊ Wait
  Waits indefinitely for a locked resource to become available. Wait is the default behavior.
- ◊ Wait=n
  The value you supply for *n* is the transaction lock timeout interval. When you

supply this value, Oracle Rdb waits *n* seconds before aborting the wait and the RMU Extract session. Specifying a wait timeout interval of zero is equivalent to specifying Nowait.

◊ Nowait
Will not wait for a locked resource to become available.

Usage Notes

- To use the RMU Extract command for a database, you must have the RMU$UNLOAD privilege in the root file access control list (ACL) for the database or the OpenVMS SYSPRV or BYPASS privilege.
- For tutorial information on using output from the RMU Extract command to load or unload a database, refer to the *Oracle Rdb Guide to Database Design and Definition*.
- Included in the output from the RMU Extract command is the SQL SET DEFAULT DATE FORMAT statement. This SQL statement determines whether columns with the DATE data type or CURRENT_TIMESTAMP builtin function are interpreted as VMS or SQL92 format. The RMU Extract command always sets the default to SQL92. The SQL92 format DATE and CURRENT_TIMESTAMP contain only the YEAR to DAY fields. The VMS format DATE and CURRENT_TIMESTAMP contain YEAR to SECOND fields.
  If your database was defined with VMS format DATE and CURRENT_TIMESTAMP, the default SQL SET DEFAULT DATE FORMAT 'SQL92' in the Extract output causes errors to be returned when you attempt to execute that output. For example, when you define a trigger:

```
SQL> CREATE TRIGGER SALARY_HISTORY_CASCADE_UPDATE
cont>     AFTER UPDATE OF JOB_CODE ON JOB_HISTORY
cont>         (UPDATE SALARY_HISTORY SH
cont>             SET SALARY_START = CURRENT_TIMESTAMP
cont>             WHERE (SH.EMPLOYEE_ID = JOB_HISTORY.EMPLOYEE_ID)
cont>         ) for each row;
%SQL-F-UNSDATASS, Unsupported date/time assignment from <Source>
 to SALARY_START
```

  You can avoid these errors by editing the output from the RMU Extract command. Replace the SET DEFAULT DATE FORMAT 'SQL92' statement with SET DEFAULT DATE FORMAT 'VMS'. If the problem occurs in trigger definitions, you can use the CAST function instead. Specify CAST(CURRENT_TIMESTAMP AS DATE VMS) with each trigger definition that references CURRENT_TIMESTAMP. (You cannot use the CAST function within the DEFAULT clause of an SQL CREATE statement).
- The following list contains a description of what the RMU Extract command generates when it encounters certain RDO statements:
  - ◊ RDO and the data dictionary have the concept of validation clauses at the domain level. The ANSI/ISO SQL92 standard allows CHECK constraints defined on domains. While the actions of the ANSI/ISO CHECK constraint do differ from VALID IF in some respects, the RMU Extract command extracts the VALID IF clauses as domain CHECK constraints if you specify the Language=SQL and Option=Full qualifiers.
  - ◊ RDO multiline descriptions
    Because the RDO interface removes blank lines in multiline descriptions, the description saved in the metadata is not identical to that entered by the database definition. The RMU Extract command therefore cannot completely reconstruct the

original description.

◊ Some RDO trigger definitions

RDO trigger definitions that contain a trigger action within a join of two or more tables generates invalid SQL syntax. For example, the following RDO trigger definition includes a join with an embedded ERASE statement. When the RMU Extract command encounters this statement, Oracle RMU generates the invalid SQL trigger definition shown.

```
DEFINE TRIGGER EXAMPLE
    AFTER ERASE
    FOR C1 IN EMPLOYEES
    EXECUTE
        FOR C2 IN JOB_HISTORY
            CROSS C3 IN EMPLOYEES
            WITH (((C2.EMPLOYEE_ID = C3.EMPLOYEE_ID)
                 AND (C2.JOB_END MISSING))
                 AND (C3.EMPLOYEE_ID = C2.EMPLOYEE_ID))
            ERASE C2
        END_FOR
    FOR EACH RECORD.

CREATE TRIGGER EXAMPLE
    AFTER DELETE ON EMPLOYEES
        (DELETE FROM JOB_HISTORY C2, EMPLOYEES C3
            WHERE (((C2.EMPLOYEE_ID = C3.EMPLOYEE_ID)
                AND (C2.JOB_END IS NULL))
                AND (C3.EMPLOYEE_ID = C2.EMPLOYEE_ID))
        ) FOR EACH ROW;
```

Note that in Oracle Rdb Version 4.1 and higher, including a trigger action within a join of two or more tables is invalid RDO syntax. For more information on this RDO restriction, see the ERASE and MODIFY entries in RDO HELP.

- Oracle CDD/Repository Version 5.3 and higher support table and column constraint definition and maintenance through CDO. The RMU Extract command, by default, assumes all constraint maintenance is with SQL and so follows each CREATE TABLE with an ALTER TABLE FROM pathname to add the constraints. However, this is no longer necessary if you are using the later versions of Oracle CDD/Repository. To disable the output of the SQL ALTER TABLE statements which add constraints use the Option=Cdd_Constraint qualifier.
- If the Transaction_Type qualifier is omitted from the RMU Extract command line, a READ ONLY transaction is started against the database. This behavior is provided for backward compatibility with prior Oracle Rdb releases. If the Transaction_Type qualifier is specified without a transaction mode, the default value Automatic is used.
- If the database has snapshots disabled and the Transaction_Type qualifier was omitted, the transaction is restarted as READ WRITE ISOLATION LEVEL READ COMMITTED to reduce the number of rows locked by operations performed with the Option=Volume_Scan qualifier enabled.
- When Transaction_Type=Write is specified, the RMU Extract process does not attempt to write to the database tables.
- In previous versions, Oracle Rdb used derived column names based on position, for example, F1, F2. With release 7.0.6.4 and later, Oracle Rdb promotes the column names from the base table into the derived column name list. The result is 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;
```

- The following list shows the equivalent SQL expressions matched by the RMU Extract process:
  - ◊ NULLIF (a, b) is eqivalent to

    ```
    CASE
      WHEN a = b THEN NULL
      ELSE a
    END
    ```
  - ◊ NVL (a, ..., b) or COALESCE (a, ..., b) is equivalent to

    ```
    CASE
      WHEN a IS NOT NULL THEN a
      ...
      ELSE b
    END
    ```
  - ◊ The simple CASE expression

    ```
    CASE a
      WHEN b THEN v1
      WHEN NULL THEN v2
      ...
      ELSE v3
    END
    ```

    is equivalent to

    ```
    CASE
      WHEN a = b THEN v1
      WHEN a IS NULL THEN v2
      ...
      ELSE v3
    END
    ```

    RMU Extract tries to decode the internal representation to as compact a SQL expression as possible.
- The RMU Extract process decodes case expressions into ABS (Absolute) functions: ABS(a) is equivalent to:

```
CASE
  WHEN a < 0 THEN −a
  ELSE a
END
```

In addition, similar forms of CASE expression are also converted to ABS:

```
CASE
  WHEN a <= 0 THEN −a
  ELSE a
END

CASE
  WHEN a > 0 THEN a
  ELSE −a
END

CASE
  WHEN a >= 0 THEN a
  ELSE −a
END
```

It is possible that the RMU Extract process will change existing CASE expressions into this more compact syntax, even if they were not originally coded as an ABS function call.

- If the Group_Table option is used and the Item qualifier omits one or more of the Table, Index, or Storage_Map keywords, only the included items are displayed. For example, to extract just the indexes for the EMPLOYEES table:

```
$ RMU/EXTRACT/ITEM=INDEX/OPTION=(GROUP_TABLE,MATCH=EMPLOYEES%)
```

To extract only the storage map and indexes for a table, use the following command:

```
$ RMU/EXTRACT/ITEM=(STORAGE_MAP,INDEX)/OPTION=(GROUP_TABLE, −
_$ MATCH=EMPLOYEES%)
```

- If the name of the LIST storage map is not known, it can be extracted using the following generic command:

```
$ RMU/EXTRACT/ITEM=STORAGE_MAP/OPTION=(GROUP_TABLE, −
_$ MATCH=RDB$SEGMENTED_STRING%)
```

Examples

Example 1

The following command extracts these database items: COLLATING_SEQUENCES, DOMAINS, TABLES, INDEXES, STORAGE_MAPS, VIEWS, and TRIGGERS.

The All option is the default. The All or Noall option can be used in conjunction with other items to select specific output. For example, the Items=(All,Nodatabase) qualifier selects all metadata items except the physical database characteristics.

```
$ RMU/EXTRACT/ITEM=(ALL, NODATABASE) MF_PERSONNEL
```

Example 2

The following command generates a DCL command procedure containing an RMU Load command for each table in the database:

```
$ RMU/EXTRACT/ITEMS=LOAD MF_PERSONNEL
```

Example 3

The following command displays the protection access control list (ACL) definitions in the mf_personnel.rdb database:

```
$ RMU/EXTRACT/ITEMS=PROTECTIONS MF_PERSONNEL.RDB
```

Example 4

The following command generates a DCL command procedure containing an RMU Unload command for each table in the database:

```
$ RMU/EXTRACT/ITEMS=UNLOAD MF_PERSONNEL.RDB
```

Example 5

The following example displays index definitions:

```
$ RMU/EXTRACT/ITEMS=INDEXES MF_PERSONNEL
```

Example 6

The following example displays domain and table definitions. Note that the Noall option could have been omitted.

```
$ RMU/EXTRACT/ITEMS=(NOALL,DOMAINS,TABLES) MF_PERSONNEL
```

Example 7

The following example displays definitions for domains (fields) and tables (relations) that reference data dictionary path names rather than using the information contained in the Oracle Rdb system tables. In addition to the database statements, it also references the data dictionary path name stored in the database, as shown in the following example:

```
$ RMU/EXTRACT/LANG=SQL/ITEM=ALL/OPTION=DIC/OUTPUT=CDD_MODEL.LOG/LOG= -
_$ CDD_EXTRACT.LOG CDD_SQL_DB
```

Example 8

The following example creates a command procedure containing a script of partial RMU Verify commands or verify command partitions for the mf_personnel database. This command procedure was created with the following RMU Extract command:

```
$ RMU/EXTRACT/ITEM=VERIFY MF_PERSONNEL
```

Example 9

The following command displays a query outline definition that was previously added to the mf_personnel database:

```
$ RMU/EXTRACT/ITEMS=(OUTLINES) MF_PERSONNEL
```

Example 10

The following command displays the after−image journal (.aij) file configuration for mf_personnel:

```
$ RMU/EXTRACT/ITEMS=(ALTER_DATABASE) MF_PERSONNEL
```

Example 11

The following command displays the function definitions in mf_personnel for functions previously created using SQL:

```
$ RMU/EXTRACT/ITEM=FUNCTION MF_PERSONNEL
```

Example 12

The following command displays the table and column cardinalities based on sorted indexes:

```
$ RMU/EXTRACT/OPTION=COLUMN_VOLUME/ITEM=VOLUME MF_PERSONNEL
```

Example 13

The following example:

- Executes an SQL EXPORT statement to create an interchange file.
- Executes an RMU Extract command with the Item=Import qualifier to generate an Import script. In addition, the Option=Filename_Only qualifier is specified to prevent full file specifications from appearing in the SQL IMPORT script. (If full file specifications are used, you cannot test the script without replacing the database that was exported.)
- Defines a logical to define the interchange file name used in the Import script file.
- Executes the Import script file.

```
SQL> -- Create interchange file, SAVED_PERS.RBR.
SQL> --
SQL> EXPORT DATABASE FILENAME MF_PERSONNEL.RDB INTO SAVED_PERS.RBR;
SQL> EXIT;
$ !
$ RMU/EXTRACT/ITEM=IMPORT/OPTION=FILENAME_ONLY/OUTPUT=IMPORT_PERS.SQL -
_$ MF_PERSONNEL
$ DEFINE/USER RMUEXTRACT_RBR SAVED_PERS.RBR
$ !
$ SQL
SQL> @IMPORT_PERS.SQL
SQL> set language ENGLISH;
SQL> set default date format 'SQL92';
SQL> set quoting rules 'SQL92';
SQL> set date format DATE 001, TIME 001;
```

```
SQL>
SQL> -- RMU/EXTRACT for Oracle Rdb V7.0-00     1-JUL-1996 15:34:38.63
SQL> --
SQL> --                          Physical Database Definition
SQL> --
SQL> ------------------------------------------------------------------------
SQL> import database from rmuextract_rbr
cont>     filename 'MF_PERSONNEL'
   .
   .
   .
```

Example 14

The following example shows an extract from the generated script when the SYS$LANGUAGE and LIB$DT_FORMAT symbols are defined. The language and format will default to ENGLISH and the standard OpenVMS format if these logical names are not defined.

```
$ DEFINE LIB$DT_FORMAT LIB$DATE_FORMAT_002,LIB$TIME_FORMAT_001
$ DEFINE SYS$LANGUAGE french
$ RMU/EXTRACT/OUT=SYS$OUTPUT/ITEM=DOMAIN MF_PERSONNEL/OPT=AUDIT_COMMENT
  .
  .
  .
-- Created on  8 janvier 2000 13:01:31.20
-- Never altered
-- Created by RDB_EXECUTE
--
SQL> CREATE DOMAIN ADDRESS_DATA_1
cont> CHAR (25)
cont> comment on domain ADDRESS_DATA_1 is
cont>   ' Street name';
  .
  .
  .
```

Example 15

If a database has snapshots set to ENABLED DEFERRED, it may be preferable to start a read/write transaction. In this environment, using the Transaction_type=(Read_only) qualifier causes a switch to a temporary snapshots ENABLED IMMEDIATE state. This transition forces 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 16

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 17

If the database currently has snapshots deferred, it may be 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. This could be explicitly stated by using the following command:

```
$ RMU/EXTRACT−
    /TRANSACTION_TYPE=(WRITE,ISOLATION=READ_COMMITTED)−
    SAMPLE.RDB
```

Using a transaction type of automatic adapts to different database settings:

```
$ RMU/EXTRACT−
    /TRANSACTION_TYPE=(AUTOMATIC)−
    SAMPLE.RDB
```

Example 18

This example shows the use of the Item=Workload qualifier to create a DCL command language script for OpenVMS systems.

```
$ RMU/EXTRACT/ITEM=WORKLOAD −
    SCRATCH/LOG/OPTION=(FILENAME,AUDIT)
$! RMU/EXTRACT for Oracle Rdb X7.0−00            7−SEP−2000 22:00:42.72
$!
$!                                WORKLOAD Procedure
$!
$!−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
$ SET VERIFY
$ SET NOON
$
$! Created on  7−SEP−2000 10:12:26.36
$! Last collected on  7−SEP−2000 22:00:34.47
$!
$ RMU/INSERT OPTIMIZER_STATISTICS −
  SCRATCH −
  /TABLE=(CUSTOMERS) −
  /COLUMN_GROUP=(CUSTOMER_NAME) −
  /DUPLICITY_FACTOR=(4.0000000) −
  /NULL_FACTOR=(0.0000000) /LOG
$
$! Created on  7−SEP−2000 10:12:26.36
$! Last collected on  7−SEP−2000 22:00:34.58
$!
$ RMU/INSERT OPTIMIZER_STATISTICS −
  SCRATCH −
  /TABLE=(RDB$FIELDS) −
  /COLUMN_GROUP=(RDB$FIELD_NAME) −
  /DUPLICITY_FACTOR=(1.7794118) −
  /NULL_FACTOR=(0.0000000) /LOG
$
   .
   .
   .
$ SET NOVERIFY
$ EXIT
```

Example 19

The following example shows the use of the Match option to select a subset of the workload entries based on the wildcard file name.

```
$ RMU/EXTRACT/ITEM=WORKLOAD -
     SCRATCH/LOG/OPTION=(FILENAME,AUDIT,MATCH:RDB$FIELDS%)
$! RMU/EXTRACT for Oracle Rdb X7.0-00                    8-SEP-2000 10:53
$!
$!                               WORKLOAD Procedure
$!
$!------------------------------------------------------------------------
$ SET VERIFY
$ SET NOON
$
$! Created on  7-SEP-2000 15:18:02.30
$! Last collected on  7-SEP-2000 18:25:04.27
$!
$ RMU/INSERT OPTIMIZER_STATISTICS -
  SCRATCH -
  /TABLE=(RDB$FIELDS) -
  /COLUMN_GROUP=(RDB$FIELD_NAME) -
  /DUPLICITY_FACTOR=(1.0000000) -
  /NULL_FACTOR=(0.0000000) /LOG
$ SET NOVERIFY
$ EXIT
```

Example 20

The following example shows the use of Item options Defer_Constraints, Constraints, and Match to extract a table and its constraints.

```
$ RMU/EXTRACT/ITEM=(TABLE,CONSTRAINT)-
_$ /OPTION=(FILENAME_ONLY,NOHEADER,-
_$          DEFER_CONSTRAINT,MATCH:EMPLOYEES%) -
_$ MF_PERSONNEL
set verify;
set language ENGLISH;
set default date format 'SQL92';
set quoting rules 'SQL92';
set date format DATE 001, TIME 001;
attach 'filename MF_PERSONNEL';
create table EMPLOYEES (
    EMPLOYEE_ID ID_DOM,
    LAST_NAME LAST_NAME_DOM,
    FIRST_NAME FIRST_NAME_DOM,
    MIDDLE_INITIAL MIDDLE_INITIAL_DOM,
    ADDRESS_DATA_1 ADDRESS_DATA_1_DOM,
    ADDRESS_DATA_2 ADDRESS_DATA_2_DOM,
    CITY CITY_DOM,
    STATE STATE_DOM,
    POSTAL_CODE POSTAL_CODE_DOM,
    SEX SEX_DOM,
    BIRTHDAY DATE_DOM,
    STATUS_CODE STATUS_CODE_DOM);
    comment on table EMPLOYEES is
      'personal information about each employee';

alter table EMPLOYEES
```

```
    add constraint EMP_SEX_VALUES
        check(EMPLOYEES.SEX in ('M', 'F', '?'))
        deferrable
    add constraint EMP_STATUS_CODE_VALUES
        check(EMPLOYEES.STATUS_CODE in ('0', '1', '2', 'N'))
        deferrable
    alter column EMPLOYEE_ID
        constraint EMPLOYEES_PRIMARY_EMPLOYEE_ID
            primary key
            deferrable;

commit work;
```

Example 21

The following example shows the use of the option Group_Table to extract a table and its indexes:

```
$ rmu/extract/item=(table,index)-
_$ /option=(group_table,match=employees%,-
_$          filename_only,noheader) db$:mf_personnel
set verify;
set language ENGLISH;
set default date format 'SQL92';
set quoting rules 'SQL92';
set date format DATE 001, TIME 001;
attach 'filename MF_PERSONNEL';
create table EMPLOYEES (
    EMPLOYEE_ID ID_DOM
        constraint EMPLOYEES_PRIMARY_EMPLOYEE_ID
            primary key
            deferrable,
    LAST_NAME LAST_NAME_DOM,
    FIRST_NAME FIRST_NAME_DOM,
    MIDDLE_INITIAL MIDDLE_INITIAL_DOM,
    ADDRESS_DATA_1 ADDRESS_DATA_1_DOM,
    ADDRESS_DATA_2 ADDRESS_DATA_2_DOM,
    CITY CITY_DOM,
    STATE STATE_DOM,
    POSTAL_CODE POSTAL_CODE_DOM,
    SEX SEX_DOM,
    BIRTHDAY DATE_DOM,
    STATUS_CODE STATUS_CODE_DOM);
    comment on table EMPLOYEES is
      'personal information about each employee';

    create unique index EMPLOYEES_HASH
        on EMPLOYEES (
        EMPLOYEE_ID)
        type is HASHED SCATTERED
        store
            using (EMPLOYEE_ID)
                in EMPIDS_LOW
                    with limit of ('00200')
                in EMPIDS_MID
                    with limit of ('00400')
                otherwise in EMPIDS_OVER;

    create unique index EMP_EMPLOYEE_ID
        on EMPLOYEES (
```

```
        EMPLOYEE_ID
            asc)
        type is SORTED
        node size 430
        disable compression;

    create index EMP_LAST_NAME
        on EMPLOYEES (
        LAST_NAME
            asc)
        type is SORTED;

commit work;

alter table EMPLOYEES
    add constraint EMP_SEX_VALUES
        check(EMPLOYEES.SEX in ('M', 'F', '?'))
        deferrable
    add constraint EMP_STATUS_CODE_VALUES
        check(EMPLOYEES.STATUS_CODE in ('0', '1', '2', 'N'))
        deferrable;

commit work;
```

Example 22

The following example shows the output when you use the Item=Revoke_Entry qualifier:

```
$ RMU/EXTRACT/ITEM=REVOKE_ENTRY ACCOUNTING_DB
...
--                              Protection Deletions
--
------------------------------------------------------------------------------

revoke entry
    on database alias RDB$DBHANDLE
    from [RDB,JAIN];

revoke entry
    on database alias RDB$DBHANDLE
    from [RDB,JONES];

revoke entry
    on database alias RDB$DBHANDLE
    from PUBLIC;

revoke entry
    on table ACCOUNT
    from [RDB,JONES];

revoke entry
    on table ACCOUNT
    from PUBLIC;

revoke entry
    on table ACCOUNT_BATCH_PROCESSING
    from [RDB,JONES];

revoke entry
```

```
    on table ACCOUNT_BATCH_PROCESSING
    from PUBLIC;
revoke entry
    on table BILL
    from [RDB,JONES];

revoke entry
    on table BILL
    from PUBLIC;
...
```

Example 23

The following example shows sample output for the WORK_STATUS table of MF_PERSONNEL. The uppercase DCL commands are generated by RMU Extract.

```
$ RMU/EXTRACT/ITEM=UNLOAD-
_$ /OPTION=(NOHEADER,FULL,MATCH:WORK_STATUS%) sql$database
$ CREATE WORK_STATUS.COLUMNS
! Columns list for table WORK_STATUS
! in DISK1:[DATABASES]MF_PERSONNEL.RDB
! Created by RMU Extract for Oracle Rdb V7.0-00 on  1-MAR-2001 20:50:25.33
STATUS_CODE
STATUS_NAME
STATUS_TYPE
$ RMU/UNLOAD -
        DISK1:[DATABASES]MF_PERSONNEL.RDB -
        /FIELDS="@WORK_STATUS.COLUMNS" -
        WORK_STATUS -
        WORK_STATUS.UNL
$
$ EXIT

$ RMU/EXTRACT/ITEM=LOAD-
_$ /OPTION=(NOHEADER,FULL,MATCH:WORK_STATUS%) sql$database
$ RMU/LOAD -
        /TRANSACTION_TYPE=EXCLUSIVE -
        /FIELDS="@WORK_STATUS.COLUMNS" -
        DISK1:[DATABASES]MF_PERSONNEL.RDB -
        WORK_STATUS -
        WORK_STATUS.UNL
$
$ EXIT
```

Example 24

The following example shows how to extract all constraints as an ALTER TABLE statement.

```
$ rmu/extract/item=(notab,constr) db$:sql_personnel/opt=(nohead,mat=empl%,defer)
set verify;
set language ENGLISH;
set default date format 'SQL92';
set quoting rules 'SQL92';
set date format DATE 001, TIME 001;
attach 'filename MF_PERSONNEL';
alter table EMPLOYEES
    add constraint EMP_SEX_VALUES
        check(EMPLOYEES.SEX in ('M', 'F', '?'))
        deferrable
```

```
    add constraint EMP_STATUS_CODE_VALUES
        check(EMPLOYEES.STATUS_CODE in ('0', '1', '2', 'N'))
        deferrable
    alter column EMPLOYEE_ID
        constraint EMPLOYEES_PRIMARY_EMPLOYEE_ID
            primary key
            deferrable;

commit work;
```

Chapter 7
Enhancements Provided in Previous Releases

7.1 Enhancements Provided in Oracle Rdb Release 7.0.7.2

7.1.1 Rdb Optional Site−Specific Startup Procedure

The Oracle Rdb startup procedure RMONSTART(xx).COM now supports an optional site−specific startup procedure to be executed after the Rdb Monitor (RDMMON) process has been started. If the file SYS$STARTUP:RDB$SYSTARTUP(xx).COM (where xx indicates the version number for multi−version Rdb kits) is found, it is executed as a DCL command procedure by the RMONSTART(xx).COM procedure.

SYS$STARTUP:RDB$SYSTARTUP(xx).COM is intended to contain site−specific tasks to be executed after the Rdb monitor procedure has completed. Such tasks might include opening databases or starting layered products that depend on the Rdb monitor process having been started.

If a site wishes to use this capability, the RDB$SYSTARTUP(xx).COM procedure must be created in SYS$STARTUP (either in the common SYS$COMMON:[SYS$STARTUP] directory or a node−specific SYS$SPECIFIC:[SYS$STARTUP] directory). The Rdb installation procedure does not provide or replace this file.

7.1.2 Oracle Rdb SGA API

Oracle Rdb maintains an extensive set of online performance statistics that provide valuable dynamic information regarding the status of an active database. The system global area (SGA) application programming interface (API) described in this document provides a way to retrieve these database performance statistics.

The SGA API automates retrieving database statistics available only through the RMU Show Statistics command. The SGA API provides the only way to retrieve statistics for Oracle Rdb databases from an application. Using the SGA API provides fast access to the data without effecting the execution of the server.

Previously, the Oracle Rdb SGA API was available as a separate software option to be downloaded, installed and maintained independently of the Oracle Rdb kit. Each time a new version of Oracle Rdb was installed, the SGA API would have to be updated. If the SGA API was not updated, it would, in many cases, fail to work correctly.

This problem has been partly resolved. Most of the contents of the SGA API separate software option are now automatically provided in the RDM$DEMO directory during the Oracle Rdb kit installation procedure. Please refer to the SGA API documentation available in RDM$DEMO in various formats as SGAAPI.PS, SGAAPI.HTML and SGAAPI.TXT for additional information.

Existing Users of the SGAAPI May Have to Modify Procedures

Existing users of the Oracle Rdb SGA API should refer to the documentation as there will be some minor changes required. In particular, the KUSRMUSHRxx.EXE sharable image is now provided in SYS$LIBRARY and the

KUSRMUSHRxx.OPT linker options file has been updated to reference this sharable image in its new location.

7.1.3 CHRONO_FLAG Replaces Older CRONO_FLAG Keyword

The SET FLAGS statement and the RDMS$SET_FLAGS logical now accept the keyword CHRONO_FLAG as an alternate spelling of the existing keyword CRONO_FLAG. The latter keyword is deprecated and will be removed from a future release of Rdb V7.1.

The new keyword will also be displayed by the SHOW FLAGS statement.

7.2 Enhancements Provided in Oracle Rdb Release 7.0.7.1

7.2.1 RDM$BIND_SNAP_QUIET_POINT Logical No Longer Used

Bug 2656534

If the logical RDM$BIND_SNAP_QUIET_POINT was defined to be "0" on a system that was used for the standby database in a Hot Standby configuration, it was not possible to start database replication. Attempts to start replication would fail with:

```
RDMS−F−HOTSNAPQUIET, quiet points must be enabled
for snapshot transactions during hot standby replication
```

However, defining the logical to "1" can cause processes with long running READ ONLY transactions to prevent database backups from proceeding.

This logical was introduced in Oracle Rdb Release 6.0 to allow database administrators to override the 6.0 requirement that READ ONLY transactions hold the quiet−point lock. Defining the logical to "1" (the default) would provide better performance when the Fast Commit feature was enabled and processes frequently switched between READ ONLY and READ WRITE transactions. Since that time, improvements have been made to the quiet−point lock algorithms that make this logical no longer necessary. Since releases 7.0.6.3 and 7.1.0.1, READ ONLY transactions would continue to hold the quiet−point lock until a backup process requested the lock. When the lock was requested, READ ONLY processes would release the quiet point lock as soon as the currently executing database request finished, if the RDM$BIND_SNAP_QUIET_POINT logical was defined as "0". This made it no longer necessary to have the logical defined as "1".

Since the quiet−point lock behavior now behaves optimally, even with the Fast Commit feature enabled, the RDM$BIND_SNAP_QUIET_POINT logical is no longer needed and thus has been removed. Oracle Rdb will now behave as if the logical is always defined to be "0".

This problem has been corrected in Oracle Rdb Release 7.0.7.1.

7.2.2 Determining Which Oracle Rdb Options Are Installed

When installing Oracle Rdb Server on OpenVMS you can choose from five components to install:

1. Oracle Rdb
2. Programmer for Rdb (Rdb Compilers)
3. Hot Standby
4. Power Utilities
5. Common Components

Starting with Rdb 7.0, you can determine what Rdb options were selected during the installation of Rdb by running the program SYS$SYSTEM:RDBINS<RdbVersionVariant>.EXE. For example:

```
$ RUN SYS$SYSTEM:RDBINS70
```

```
Installed: Oracle Rdb,Rdb Compilers,Hot Standby,Power Utilities
```

Previously, however, the output of the RDBINS program could not easily be redirected. Attempts to redefine SYS$OUTPUT would not allow the program output to be captured.

This problem has been resolved. The RDBINS program now allows redirection of the output to SYS$OUTPUT. The RDBINS program also creates a DCL symbol RDB$INSTALLED_SELECTIONS containing the same output string as is displayed to SYS$OUTPUT.

7.2.3 New Procedure RDB$IMAGE_VERSIONS.COM

The command procedure RDB$IMAGE_VERSIONS.COM is supplied in SYS$LIBRARY by the Rdb installation procedure. The RDB$IMAGE_VERSIONS command procedure can be used to display the image identification string and image link date/time from various Oracle Rdb or potentially related images in SYS$SYSTEM, SYS$LIBRARY and SYS$MESSAGE. This procedure can be used to determine exactly what images are installed on the system.

RDB$IMAGE_VERSIONS.COM accepts an optional parameter. If passed, this parameter specifies a specific file or wildcard to lookup and display information for. By default, filenames starting with RD*, SQL*, RM*, and COSI* and ending with .EXE are searched for and displayed.

The following example shows how to use the RDB$IMAGE_VERSIONS command procedure.

```
Decrdb RTA1:> @RDB$IMAGE_VERSIONS
SYS$SYSROOT:[SYSEXE]RDB$NATCONN71.EXE;1 SQL*NET V7.1-55   8-MAY-2002 15:56
SYS$COMMON:[SYSEXE]RDBINS.EXE;4         ORACLE RDB V7.0  14-NOV-2002 17:20
SYS$COMMON:[SYSEXE]RDBINS70.EXE;33      ORACLE RDB V7.0   7-MAR-2003 15:30
SYS$COMMON:[SYSEXE]RDBINS71.EXE;5       ORACLE RDB V7.1   9-APR-2003 10:58
SYS$COMMON:[SYSEXE]RDBPRE.EXE;5         V7.0-65          10-SEP-2002 16:02
SYS$COMMON:[SYSEXE]RDBPRE70.EXE;37      V7.0-7           28-FEB-2003 23:24
SYS$COMMON:[SYSEXE]RDBPRE71.EXE;5       V7.1-101          8-APR-2003 16:49
SYS$COMMON:[SYSEXE]RDBSERVER.EXE;9      RDB/RSV V7.0-65   5-SEP-2002 21:01
SYS$COMMON:[SYSEXE]RDBSERVER70.EXE;41   RDB/RSV V7.0-7   27-FEB-2003 17:29
SYS$COMMON:[SYSEXE]RDBSERVER71.EXE;5    RDB/RSVV7.1-101   7-APR-2003 17:43
SYS$COMMON:[SYSEXE]RDMABS.EXE;5         RDB V7.0-65      10-SEP-2002 16:01
    .
    .
    .
```

Chapter 8
Documentation Corrections

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

8.1 Documentation Corrections

8.1.1 Database Server Process Priority Clarification

By default, the database servers (ABS, ALS, DBR, LCS, LRS, RCS) created by the Rdb monitor inherit their VMS process scheduling base priority from the Rdb monitor process. The default priority for the Rdb monitor process is 15.

Individual server priorities can be explicitly controlled via system−wide logical names as described in Table 8−1.

Table 8−1 Server Process Priority Logical Names

Logical Name	Use
RDM$BIND_ABS_PRIORITY	Base Priority for the ABS Server process
RDM$BIND_ALS_PRIORITY	Base Priority for the ALS Server process
RDM$BIND_DBR_PRIORITY	Base Priority for the DBR Server process
RDM$BIND_LCS_PRIORITY	Base Priority for the LCS Server process
RDM$BIND_LRS_PRIORITY	Base Priority for the LRS Server process
RDM$BIND_RCS_PRIORITY	Base Priority for the RCS Server process

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

For most applications and systems, Oracle discourages changing the server process priorities.

8.1.2 Waiting for Client Lock Message

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

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

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

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

The leftmost value seen in the hexadecimal output contains the id of the object. The id is described below for tables and views, functions, procedures, and modules.

- For tables and views, the id represents the unique value found in the RDB$RELATION_ID column of the RDB$RELATIONS system relation for the given table.
- For routines, the id represents the unique value found in the RDB$ROUTINE_ID column of the RDB$ROUTINES system relation for the given routine.
- For modules, the id represents the unique value found in the RDB$MODULE_ID column of the RDB$MODULES system relation for the given module.

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

Table 8−2 Objects and Their Hexadecimal Type Value

Object	Hexadecimal Value
Tables or views	00000004
Routines	00000016
Modules	00000015

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

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

```
Process.ID Since.....   Stall.reason.......................... Lock.ID.
46001105:2 10:40:46.38 − waiting for client '........' 0000001900000000400000055
```

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;
```

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:

- Press the L option from the horizontal menu to display a menu of lock IDs.
- Select the desired lock ID.

8.1.3 Clarification of PREPARE Statement Behavior

Bug 2581863

According to the Oracle Rdb7 SQL Reference Manual, Volume 3 page 7−227, when using a statement−id parameter for PREPARE "if that parameter is an integer, then you must explicitly initialize that integer to zero before executing the PREPARE statement".

This description is not correct and should be replaced with this information:

1. If the statement−id is non−zero and does not match any prepared statement (the id was stale or contained a random value), then an error is raised:
   %SQL−F−BADPREPARE, Cannot use DESCRIBE or EXECUTE on a statement that is not prepared
2. If the statement−id is non−zero, or the statement name is one that has previously been used and matches an existing prepared statement, then that statement is automatically released prior to the prepare of the new statement. Please refer to the RELEASE statement for further details.
3. If the statement−id is zero or was automatically released, then a new statement−id is allocated and the statement prepared.

Please note that if you use statement−name instead of a statement−id−parameter then SQL will implicitly declare an id for use by the application. Therefore, the semantics described apply similarly when using the statement−name. See the RELEASE statement for details.

8.1.4 SQL EXPORT Does Not Save Some Database Attributes

Bug 2574640

The SQL EXPORT and IMPORT commands do not support several database attributes.

The following attributes are not saved by EXPORT for this release of Oracle Rdb.

- CHECKPOINT TIMED EVERY n SECONDS
- CHECKPOINT { ALL | UPDATED } ROWS TO { BACKING FILE | DATABASE }
- RMU/SET LOGMINER

This problem has been corrected in Oracle Rdb Release 7.1. The EXPORT protocol has been extended to support these and other database attributes. If you have databases with these attributes enabled then after the IMPORT completes, use a SQL script to re−establish the settings for the new database.

8.1.5 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.

8.1.6 New Request Options for RDO, RDBPRE and RDB$INTERPRET

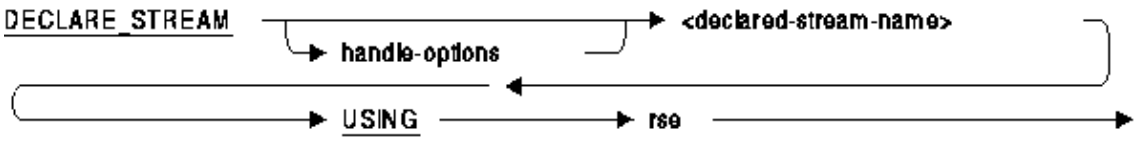
This release note was included in the V70A Release Notes but had gotten dropped somewhere along the line.

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.

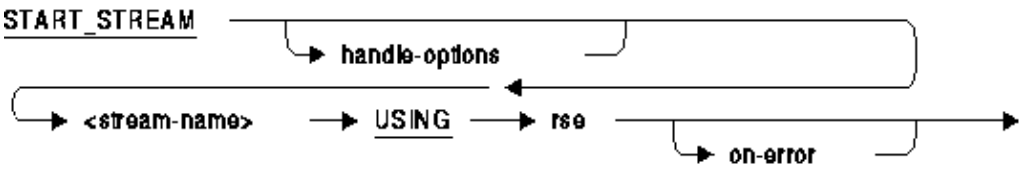
In prior releases, the handle−options could not be specified in interactive RDO or RDB$INTERPRET. This has changed in Rdb but these allowed options will be limited to MODIFY and PROTECTED keywords. For RDBPRE, all options listed will be supported. These option names were chosen to be existing keywords 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.

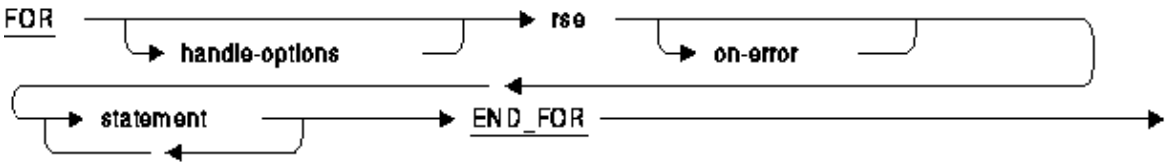
Example 5−1 DECLARE_STREAM Format



Example 5−2 START_STREAM Format

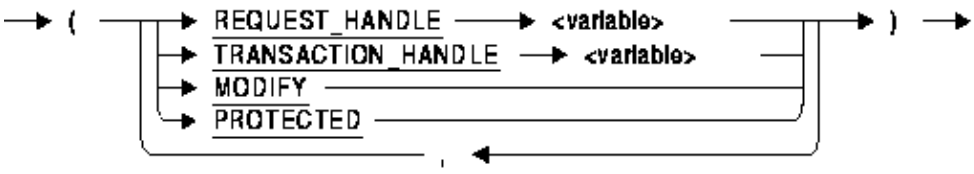


Example 5−3 FOR Format



Each of these statements references the syntax for the HANDLE−OPTIONS which has been revised and is shown below.

handle-options =



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.

8.1.7 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 8−3 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 ainteger 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 8−4 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 8−5 Changed Columns for RDB$ROUTINES Table

Column	Data	Domain Name	Comments

Name	Type		
RDB$FLAGS	Integer	RDB$FLAGS	A bit mask where the bits have the following meaning when set:
			Bit 0: Routine is a function. (call returns a result.)
			Bit 1: Routine is not valid. (Invalidated by a metadata change.)
			Bit 2: Function is not deterministic (that is, the routine is variant). A subsequent invocation of the routine with identical parameters may return different results.
			Bit 3: Routine can change the transaction state.
			Bit 4: Routine is in a secured shareable image.
			Bit 5: Reserved for future use.
			Bit 6: Routine is not valid. (Invalidated by a metadata change to the object upon which this routine depends. This dependency is a language semantics dependency.)
			Bit 7: Reserved for future use.
			Bit 8: External function returns NULL when called with any NULL parameter.
			Bit 9: Routine has been analyzed (used for trigger dependency tracking).
			Bit 10: Routine inserts rows.
			Bit 11: Routine modifies rows.
			Bit 12: Routine deletes rows.
			Bit 13: Routine selects rows.
			Bit 14: Routine calls other routines.
			Other bits are reserved for future use.

Table 8−6 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.

8.1.8 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
```

8.1.9 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
```

8.1.10 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.

8.1.11 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.

8.1.12 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.

8.1.13 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 HP 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
```

8.1.14 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.

8.1.15 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 HP 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.

8.1.16 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;
```

8.1.17 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.

8.1.18 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*.

8.1.19 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.

8.1.20 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*.

8.1.21 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.

8.1.22 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
```

8.1.23 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.

8.1.24 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
```

8.1.25 Undocumented SET Commands and Language Options

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

8.1.25.1 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;
```

8.1.25.2 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;
```

8.1.26 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
```

8.1.27 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.

8.1.28 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.

8.1.29 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.

8.1.30 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 7.0.8.

9.1 Oracle Rdb Considerations

9.1.1 RDMS−E−RTNSBC_INITERR, Cannot Init External Routine Server Site Executor

Execution of an external function (or procedure) with server site binding may unexpectedly fail.

The following message is an example of the error message you might see:

```
%RDB−E−EXTFUN_FAIL, external routine failed to compile or execute successfully
−RDMS−E−EXTABORT, routine NNNNNNNNN execution has been aborted
−RDMS−E−RTNSBC_INITERR, Cannot init. external routine server site executor;
reason XX
```

Where NNNNNNNNN is the function name and XX is a decimal value, e.g., 41.

While such errors are possible they are very unlikely to be seen, especially on systems that have had Oracle Rdb successfully installed. These errors usually indicate a problem with the environment. For instance, ensure that images RDMXSMvv.EXE, RDMXSRvv.EXE and RDMXSMPvv.EXE (where vv is the Rdb version) are installed and have the correct protections:

```
Directory DISK$:<SYS6.SYSCOMMON.SYSLIB>

RDMXSM70.EXE;3           183   8−APR−2004 09:37:31.36  (RWED,RWED,RWED,RE)
RDMXSMP70.EXE;3          159   8−APR−2004 09:37:31.54  (RWED,RWED,RWED,RE)
RDMXSR70.EXE;3            67   8−APR−2004 09:37:31.74  (RWED,RWED,RWED,RE)

Total of 3 files, 409 blocks.

DISK$:<SYS6.SYSCOMMON.SYSLIB>.EXE
   RDMXSM70;3       Open Hdr Shared            Lnkbl
DISK$:<SYS6.SYSCOMMON.SYSLIB>.EXE
   RDMXSMP70;3      Open Hdr Shared       Prot Lnkbl  Safe
DISK$:<SYS6.SYSCOMMON.SYSLIB>.EXE
   RDMXSR70;3       Open Hdr Shared            Lnkbl
```

9.1.2 AIJ Log Server Process May Loop Or Bugcheck

Bugs 2651475 and 1756433

Under unknown, but extremely rare conditions, on busy databases where the After Image Journal (AIJ) Log Server process is enabled, the ALS process has been observed to enter a loop condition writing AIJ information to the AIJ file(s).

In the worst case, this problem could cause all available journal files to be filled with repeating data. If no remedial action were to be taken, this condition could cause the database to be shutdown and the AIJ journals to be considered inaccessable.

The database is not corrupted by this problem.

Stopping and restarting the ALS process will clear the looping condition even if the ALS process must be stopped using the STOP/ID command.

Stopping the ALS process will not impact production as AIJ writes will automatically revert to the non−ALS behaviour.

In Oracle Rdb Release 7.0.7 the behaviour of Rdb has been changed so that should this problem be detected, the ALS process will automatically shutdown producing a bugcheck dump file. This will prevent any danger of filling all available journals and will ensure that the database remains available.

ALS may be safely restarted immediately as the conditions that cause such a loop are resolved during recovery of the ALS process.

9.1.3 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.

9.1.4 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*.

9.1.5 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.

9.1.6 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.

9.1.7 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. This feature is available in Rdb Version 7.1.

9.1.8 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.

9.1.9 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".

9.1.10 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.

9.1.11 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.

9.1.12 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.

9.1.13 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.

9.1.14 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 Rdb V7.1 release, support is provided to inherit the users full security profile into the database.

9.1.15 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.

9.1.16 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.

9.1.17 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.

9.1.18 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 is lifted in the Rdb V7.1 releases. 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.

9.1.19 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.

9.1.20 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.

9.1.21 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.

9.1.22 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.

9.1.23 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.

9.1.24 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.

9.1.25 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.

9.1.26 Oracle Rdb and the SRM_CHECK Tool

The Alpha Architecture Reference Manual, Third Edition (AARM) describes strict rules for using interlocked memory instructions. The HP 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 HP 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 HP 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 HP Alpha 21264 (EV6) processor interlocked memory instructions issues is available at:

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

9.1.27 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.

9.1.28 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.

9.1.29 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.

9.1.30 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.

9.1.31 DBAPack for Windows 3.1 is Deprecated

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

9.1.32 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.

9.1.33 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.

9.1.34 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.

9.1.35 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.

9.1.36 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.

9.1.37 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.

9.1.38 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
```

9.1.39 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.

9.1.40 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.

9.1.41 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.

9.1.42 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.

9.1.43 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.

9.1.44 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.

9.1.45 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
```

9.1.46 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);
```

9.1.47 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.

9.1.48 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.

9.1.49 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.

9.1.50 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.

9.1.51 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 HP Computer Corporation representative for information about patches to DECdtm.

9.1.52 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*.

9.1.53 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.

9.1.54 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.

9.1.55 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.

9.1.56 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.)

9.1.57 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.

9.1.58 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

9.1.59 Side Effect when Calling Stored Routines

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

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

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

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

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

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

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

9.1.60 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.

9.1.61 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.

9.1.62 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;
```

9.1.63 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.

9.2 Oracle CDD/Repository Restrictions

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

9.2.1 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.

9.2.2 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.

9.2.3 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 9.2.3.1 for details.

9.2.3.1 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.

9.2.3.2 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