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

Oracle® Rdb for OpenVMS

Release Notes

Release 7.1.5.2

October 2007

Oracle Rdb Release Notes, Release 7.1.5.2 for OpenVMS

Contents

Preface

Purpose of This Manual

This manual contains release notes for Oracle Rdb Release 7.1.5.2. The notes describe changed and enhanced features; upgrade and compatibility information; new and existing software problems and restrictions; and software and documentation corrections.

Intended Audience

This manual is intended for use by all Oracle Rdb users. Read this manual before you install, upgrade, or use Oracle Rdb Release 7.1.5.2.

Document Structure

This manual consists of the following chapters:

Chapter 1	Describes how to install Oracle Rdb Release 7.1.5.2.
Chapter 2	Describes software errors corrected in Oracle Rdb Release 7.1.5.2.
Chapter 3	Describes software errors corrected in Oracle Rdb Release 7.1.5.1.
Chapter 4	Describes software errors corrected in Oracle Rdb Release 7.1.5.
Chapter 5	Describes enhancements introduced in Oracle Rdb Release 7.1.5.1.
Chapter 6	Describes enhancements introduced in Oracle Rdb Release 7.1.5.
Chapter 7	Provides information not currently available in the Oracle Rdb documentation set.
Chapter 8	Describes problems, restrictions, and workarounds known to exist in Oracle Rdb Release 7.1.5.2.

Chapter 1
Installing Oracle Rdb Release 7.1.5.2

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

NOTE

All Oracle Rdb Release 7.1 kits are full kits. There is no requirement to install any prior release of Oracle Rdb when installing new Rdb Release 7.1 kits.

1.1 Oracle Rdb V7.1 Version Numbering Enhancement

Previously, the Oracle Rdb version number was specified as 4 digits (for example, version "7.1.0.2"). Starting with Oracle Rdb Release 7.1.1, an additional, fifth, digit has been added to the kit version number. This new digit is intended to indicate an optimization level of the Rdb software. The use of this new digit is to indicate a "generic" kit (final digit of zero) for all Alpha processors or a "performance" kit that will run on a subset of the supported platforms (final digit of 1). In the future, additional values may be specified to indicate other performance or platform options.

For Oracle Rdb Release 7.1.5.2, the two kits are 7.1.5.2.0 (compiled for all Alpha processor types) and 7.1.5.2.1 (compiled for EV56 and later Alpha processors). These kits offer identical functionality and differ only in a potential performance difference.

1.2 Requirements

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

- Oracle Rdb must be shutdown before you install this update kit. That is, the command file SYS$STARTUP:RMONSTOP71.COM should be executed before proceeding with this installation. If you have an OpenVMS cluster, you must shutdown the Rdb 7.1 monitor on all nodes in the cluster before proceeding.
- The installation requires approximately 280,000 blocks for OpenVMS Alpha systems.
- If you are running Hot Standby and you are upgrading from a version of Oracle Rdb 7.1 prior to 7.1.1, you must install this kit on both the master and the standby systems prior to restarting Hot Standby. This requirement is necessary due to changes to the message format used to transmit journal state information from the master to the standby system.

1.3 Invoking VMSINSTAL

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

To install the Oracle Rdb for OpenVMS Alpha kit that is compiled to run on all Alpha platforms:

```
@SYS$UPDATE:VMSINSTAL RDBV71520AM device−name OPTIONS N
```

To install the Oracle Rdb for OpenVMS Alpha kit that is performance targeted for Alpha EV56 and later platforms:

```
@SYS$UPDATE:VMSINSTAL RDBV71521AM device−name OPTIONS N
```

device−name

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

OPTIONS N

This parameter prints the release notes.

The full Oracle Rdb Release 7.1 Installation Guide is also available on MetaLink in Adobe Acrobat PDF format:

```
Top Tech Docs\Oracle Rdb\Documentation\Rdb 7.1 Installation and Configuration
Guide
```

1.4 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.

1.5 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, "RDBVMSV7.1".

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=RDBVMSV7.1 −
_$ /OUT=RDBVMS.EPC$DEF SYS$SHARE:EPC$FACILITY.TLB
```

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

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

Note that the process executing the INSERT DEFINITION command must use the version of Oracle Rdb that matches the version used to create the Oracle TRACE administration database or the INSERT DEFINITION command will fail.

1.6 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>RDMXSMP71.EXE
−INSTALL−E−SYSVERDIF, system version mismatch − please relink
```

This message may be safely ignored. Using the RMONSTOP.COM (for standard installations) or RMONSTOP71.COM (for multi−version installations) procedure to shut down Oracle Rdb prior to the installation may help avoid the message.

1.7 Patches for OpenVMS V7.3−1

Several problems that affect installations using Oracle Rdb on OpenVMS V7.3−1 are corrected in patch kits available from HP OpenVMS support. Oracle recommends that you consult with Hewlett−Packard and install these patch kits (or their replacements) to correct or avoid the following problems:

- VMS731_SYS−V0400 corrects the following problems seen with Oracle Rdb:
  - When using Oracle Rdb Galaxy support, or memory−resident global sections, processes enter a permanent RWAST state at image exit. The system must be rebooted to remove the process and continue normal operations. Note that when using Oracle Rdb Release 7.1.2 databases with SHARED MEMORY IS PROCESS RESIDENT attribute, the Row Cache feature and caches with the SHARED MEMORY IS SYSTEM, LARGE MEMORY IS ENABLED, or RESIDENT attributes, or in an OpenVMS Galaxy configuration with Oracle Rdb Galaxy support enabled, you are at an elevated risk of experiencing this problem.
    Configurations that do not have this patch, or it's future replacement, applied will not be supported by Oracle if the SHARED MEMORY IS PROCESS RESIDENT, the Row Cache, or Galaxy support features are in use. If you are not using these features, then the patch or it's replacement is not mandatory. However, Oracle still strongly recommends that it be used.
  - Applications using the Oracle Rdb Row Cache or AIJ Log Server (ALS) features would sometimes have their server processes hang in HIB (hibernate) state.
- VMS731_SYSLOA−V0100 corrects the following problem seen with Oracle Rdb:
  - In an OpenVMS cluster environment, unreported deadlocks and hangs can occur. This problem is sometimes characterized by an Oracle Rdb blocking lock incorrectly being shown as owned by the system (in other words, with a zero PID).

1.8 Oracle Rdb Release 7.1.5.2.1 Optimized for Alpha EV56 (21164A Processor Chip) and Later Platforms

Oracle will be releasing Oracle Rdb 7.1 and later kits in parallel build streams − a "generic" kit that will run on all certified and supported Alpha platforms as well as a "performance" kit that will run on a subset of the supported platforms. The performance kit is intended for those customers with "newer" Alpha processor chips who need higher levels of performance than are offered by the generic kits. The performance kits are otherwise functionally identical to the generic kits.

Oracle will continue to release both types of kits for Oracle Rdb Release 7.1 as long as there is significant customer interest in the generic kit.

For improved performance on current generation Alpha processors, Oracle Rdb Release 7.1.5.2.1 is compiled explicitly for Alpha EV56 and later systems. This version of Oracle Rdb requires a system with a minimum Alpha processor chip of EV56 and a maximum processor chip of Alpha EV7 (known as the Alpha 21364).

Oracle Rdb Release 7.1.5.2.1 is functionally equivalent to Oracle Rdb Release 7.1.5.2.0 and was built from the same source code. The only difference is a potentially improved level of performance. Oracle Rdb Releases 7.1.5.2.0 and 7.1.5.2.1 are certified on all supported Alpha processor types (up to and including the Alpha EV7 processor).

In Release 7.1.5.2.1, Oracle Rdb is explicitly compiled for EV56 and later Alpha processors such that the generated instruction stream can utilize the byte/word extension (BWX) of the Alpha architecture. Additionally, this kit is compiled with instruction tuning biased for performance of Alpha EV6 and later systems that support quad−issue instruction scheduling.

Note that you should not install Release 7.1.5.2.1 of Oracle Rdb on Alpha EV4, EV45 or EV5 systems. These processor types do not support the required byte/word extension (BWX) of the Alpha architecture. Also ensure that all systems in a cluster sharing the system disk are using a minimum of the Alpha EV56 processor.

To easily determine the processor type of a running OpenVMS Alpha system, use the CLUE CONFIG command of the OpenVMS System Dump Analyzer utility (accessed with the ANALYZE/SYSTEM command). The "CPU TYPE" field indicates the processor type as demonstrated in the following example from an HP AlphaServer GS140 6/525 system with an EV6 (21264) processor:

```
$ ANALYZE/SYSTEM
SDA> CLUE CONFIG
System Configuration:
    .
    .
    .
Per−CPU Slot Processor Information:
CPU ID    00                  CPU State    rc,pa,pp,cv,pv,pmv,pl
CPU Type  EV6  Pass 2.3 (21264)
PAL Code  1.96−1              Halt PC      00000000.20000000
    .
    .
    .
```

1.8.1 AlphaServer 4000 EV56 299Mhz Not Supported by Oracle Rdb Release Optimized for Alpha EV56 Processor

Oracle Rdb releases that are optimized for the Alpha EV56 and later processors are not able to run on the AlphaServer 4000 with the 299Mhz EV56 processor. Though this CPU claims to be an EV56, it does not, in fact, implement the byte/word instruction set as required.

According to information on the HP web site, this problem may be present in the AlphaServer 4000 or 4100 systems with a processor module of KN304−FA or KN304−FB. The systems effected appear to include the AlphaServer 4x00 5/300 pedestal, cabinet and rackmount systems: DA−51FAB−ED/−FD/−GB or DA−53GEB−CA/−EA/−FA/−GA.

The indicated CPU is not able to run Oracle Rdb releases that are optimized for the Alpha EV56 and later processors. This effects Oracle Rdb Releases optimized for the Alpha EV56 and later processors.

Possible workarounds include updating the system to an EV56 module for the AlphaServer 4x00 that is later than the KN304−FA or FB (ie a clock speed greater than 300Mhz). Some of the possible modules would include: KN304−AA 400mhz, KN304−DA 466mhz, B3005−CA 533mhz, B3006−EB 600mhz.

Otherwise, an Oracle Rdb release that is not optimized for the Alpha EV56 and later processors must be used (such as Oracle Rdb Releases 7.1.5.2.0)

Please contact your HP AlphaServer hardware vendor for additional information.

1.9 Maximum OpenVMS Version Check Added

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

OpenVMS Version 8.3−x is the maximum supported version of OpenVMS.

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

1.10 VMS$MEM_RESIDENT_USER Rights Identifier Required

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

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

Chapter 2
Software Errors Fixed in Oracle Rdb Release 7.1.5.2

This chapter describes software errors that are fixed by Oracle Rdb Release 7.1.5.2.

2.1 Software Errors Fixed That Apply to All Interfaces

2.1.1 Potential System Crash Using VLM Global Buffers

In very rare conditions, when using the Global Buffers VLM (Very Large Memory) feature, it was possible for Oracle Rdb to cause an OpenVMS system crash with a bugcheck type of "MACHINECHK, Machine check while in kernel mode". The MACHINECHK can be triggered by an invalid PFN (page frame number) value being loaded into a PTE (page table entry).

The workaround to this problem is to discontinue use of the Oracle Rdb Global Buffers VLM (Very Large Memory) feature by specifying "ALTER DATABASE ... GLOBAL BUFFERS ...LARGE MEMORY IS DISABLED".

This problem has been corrected in Oracle Rdb Release 7.1.5.2. The Oracle Rdb VLM feature no longer can access past the end of the allocated PFN map.

2.1.2 Bugcheck From UNION Query With Constant Boolean

Bug 6125013

It was reported that a bugcheck was received from a UNION query with a constant boolean, as shown in the following example.

```
SELECT A_CLEARER_ID
FROM (SELECT A_DATE,A_CLEARER_ID
       FROM   T_CLEARED_BY
       UNION ALL
       SELECT A_DATE,A_CLEARER_ID
       FROM   T_CLEARED_BY
) AS   TMP
WHERE  A_DATE = DATE'2007-06-15' AND 1=1;
%RDMS-I-BUGCHKDMP, generating bugcheck dump file DISK:[DIRECTORY]RDSBUGCHK.DMP;
```

The query works if the constant predicate "1=1" is removed:

```
SELECT A_CLEARER_ID
FROM (SELECT A_DATE,A_CLEARER_ID
       FROM   T_CLEARED_BY
       UNION ALL
       SELECT A_DATE,A_CLEARER_ID
       FROM   T_CLEARED_BY
) AS   TMP
WHERE  A_DATE = DATE'2007-06-15';
Merge of 1 entries
  Merge block entry 1
  Merge of 2 entries
    Merge block entry 1
    Conjunct        Index only retrieval of relation T_CLEARED_BY
      Index name  U1_CLEARED_BY [1:1]
    Merge block entry 2
    Conjunct        Index only retrieval of relation T_CLEARED_BY
```

```
      Index name  U1_CLEARED_BY [1:1]
0 rows selected
```

This problem occurs when the main query selects from a derived table of a union query with a boolean predicate such as "1=1" in the WHERE clause.

This problem affects the following Oracle Rdb versions: V7.1.4.4, V7.1.4.5, V7.1.5, V7.1.5.1, V7.2.0.1, V7.2.0.2 and V7.2.1, V7.2.1.1 and V7.2.1.2. This problem has been corrected in Oracle Rdb Release 7.1.5.2.

2.1.3 CPU Bound Loop in PIOUTL$SERVICE_PAGE_BLASTS

Bug 6111009

The ACMS server process gets stuck in a CPU bound loop. PC samples taken from the process or from a forced bugcheck dump point to locations in routine PIOUTL$SERVICE_PAGE_BLASTS.

The problem was a missed synchronization to prevent delivery of blocking ASTs. The outcome was the code inserted elements (BCBs) twice into a queue (BLASTED_QUEUE) causing a loop in the queue linkage.

The problem has been corrected in Oracle Rdb Release 7.1.5.2. A new flag (in the BCB) has been added together with interlocked test instructions to prevent this situation.

2.1.4 Wrong Result From Zigzag Match

Bug 6126475

A query with zigzag match strategy returns the wrong result (using the outline bug_match_outline which applies the index T1_SC_PC_NDX at the outer leg).

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

SELECT E.PCODE, P.PKEY
  FROM T1 E,
       T2 P
 WHERE E.SCODE = 'AXA'
   AND P.SNO = 4
   AND E.SCODE = P.SCODE
   AND E.PKEY = P.PKEY  ;
~S: Outline "BUG_MATCH_OUTLINE" used
~S#0003
Tables:
  0 = T1
  1 = T2
Conjunct: (0.SCODE = 1.SCODE) AND (0.PKEY = 1.PKEY)
Match
  Outer loop
    Sort: 0.SCODE(a), 0.PKEY(a)
      SortId# 4., # Keys 4
        Item# 1, Dtype: 2, Order: 0, Off: 0, Len: 1
        Item# 2, Dtype: 14, Order: 0, Off: 1, Len: 3
        Item# 3, Dtype: 2, Order: 0, Off: 4, Len: 1
        Item# 4, Dtype: 8, Order: 0, Off: 5, Len: 4
        LRL: 32, NoDups:0, Blks:6, EqlKey:0, WkFls: 2
```

```
    Leaf#01 BgrOnly 0:T1 Card=3
      Bool: 0.SCODE = 'AXA'
      BgrNdx1 T1_SC_PC_NDX [1:1] Fan=16
        Keys: 0.SCODE = 'AXA'
  Inner loop      (zig-zag)
    Index only retrieval of relation 1:T2
      Index name  T2_SC_SN_PK_NDX [2:2]
        Keys: (1.SCODE = 'AXA') AND (1.SNO = 4)
0 rows selected
```

The query works if the outline is changed to use the index T1_SC_PK_PC_NDX instead of T1_SC_PC_NDX.

```
create outline BUG_MATCH_OUTLINE
id '045E2C384D284FD1061D1BF58CF8872D'
mode 0
as (
  query (
-- For loop
    subquery (
      T1 0         access path index
--    T1_SC_PC_NDX        -- replace it with T1_SC_PK_PC_NDX
      T1_SC_PK_PC_NDX
        join by match to
      T2 1      access path index
      T2_SC_SN_PK_NDX
      )
    )
  )
compliance optional     ;

SELECT E.PCODE, P.PKEY
  FROM T1 E,
       T2 P
 WHERE E.SCODE = 'AXA'
   AND P.SNO = 4
   AND E.SCODE = P.SCODE
   AND E.PKEY = P.PKEY  ;
~S: Outline "BUG_MATCH_OUTLINE" used
Tables:
  0 = T1
  1 = T2
Conjunct: (0.SCODE = 1.SCODE) AND (0.PKEY = 1.PKEY)
Match
  Outer loop      (zig-zag)
    Index only retrieval of relation 0:T1
      Index name  T1_SC_PK_PC_NDX [1:1]
        Keys: 0.SCODE = 'AXA'
  Inner loop      (zig-zag)
    Index only retrieval of relation 1:T2
      Index name  T2_SC_SN_PK_NDX [2:2]
        Keys: (1.SCODE = 'AXA') AND (1.SNO = 4)
 E.PCODE        P.PKEY
     850         84900
     850         84910
2 rows selected
```

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

1. The main query joins two tables using a one−sided zigzag match strategy where the zigzag skip

occurs only at the outer leg.

2. One of the match join keys (i.e. PKEY) is NOT included in the outer index and thus a sort is applied to produce the required index order at the outer leg.

This problem has been corrected in Oracle Rdb Release 7.1.5.2.

2.1.5 Wrong Result From Outer Join With MISSING Predicate

Bug 6404754

Wrong results were reported from an outer join query with a "Missing" predicate. The customer's query is simplified into the following reproducer:

```
create table TAB1 (COL1 CHAR(4));
create table TAB2 (COL1 CHAR(4));

insert into TAB1 values ('1030');
insert into TAB1 values ('1420');

insert into TAB2 values ('1030');
insert into TAB2 values ('1420');

commit;

! the following query should return 0 rows but returns 2 rows incorrectly:
!
SELECT *
  FROM TAB1 AAA,
 (SELECT TAB1.COL1 FROM TAB1, TAB2 WHERE TAB1.COL1 = TAB2.COL1(+) ) BBB
   WHERE AAA.COL1 = BBB.COL1(+) AND BBB.COL1 IS NULL;
Tables:
  0 = TAB1
  1 = TAB1
  2 = TAB2
Conjunct: MISSING (1.COL1)
Cross block of 2 entries        (Left Outer Join)
  Cross block entry 1
    Get     Retrieval sequentially of relation 0:TAB1
  Cross block entry 2
    Conjunct: MISSING (1.COL1)         <== See NOTE below
    Conjunct: 0.COL1 = 1.COL1
    Merge of 1 entries
      Merge block entry 1
      Conjunct: MISSING (1.COL1)        <== See NOTE below
      Cross block of 2 entries        (Left Outer Join)
        Cross block entry 1
          Get     Retrieval sequentially of relation 1:TAB1
        Cross block entry 2
          Conjunct: 1.COL1 = 2.COL1
          Get     Retrieval sequentially of relation 2:TAB2
 AAA.COL1   BBB.COL1
 1030       NULL
 1420       NULL
2 rows selected
```

NOTE:: The conjunct is incorrectly pushed down from the main query across the left outer join query and thus causing the wrong result.

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

1. The main query is a left outer join query between a table and a derived table.
2. The derived table is derived from another left outer join query of two tables.
3. The WHERE clause contains an IS NULL predicate referencing the join column of the derived table.

This problem has been corrected in Oracle Rdb Release 7.1.5.2.

2.1.6 Query Bugchecks After Upgrading From Release 7.1.1 to Release 7.1.5.1

Bug 6448202

A query bugchecks after a customer upgrades from Release 7.1.1 to Release 7.1.5.1. The query joins several tables with three other complicated views.

The query succeeds if the dynamic optimizer is disabled by defining the following SQL flags:

```
set flags 'max_stability';
```

This problem was introduced in Oracle Rdb Release 7.1.4.2 by a fix for Bug 4597186 but no problem has been encountered until the recent customer report. One of the retrieval blocks under a multiply nested cross strategy applied the dynamic optimizer LEAF node by walking up to the top parent query.

This problem has been corrected in Oracle Rdb Release 7.1.5.2.

2.1.7 Query Loops and Hangs Upgrading from Rdb V7.0 to V7.2

Bug 6459494

The query loops and hangs after the customer upgrades from Rdb Release 7.0.6 to an Rdb 7.2 release. The following query is a simplified one that displays the same behavior (very slow performance):

```
set flags 'strategy';
SELECT count(*)
     FROM PM P, KOSE K
     WHERE
         P.PROD = 'RSS' AND
         EXISTS(SELECT *
                 FROM KSU_V KV
                 WHERE
                     P.PROD = KV.PROD AND
                     P.PROD = K.PROD );
Aggregate
Cross block of 2 entries
  Cross block entry 1
    Match
      Outer loop      (zig-zag)
```

```
        Index only retrieval of relation KOSE
          Index name  IDX_KOSE [1:1]
      Inner loop      (zig-zag)
        Index only retrieval of relation PM
          Index name  IDX_PM [1:1]
  Cross block entry 2
    Conjunct        Aggregate-F1    Conjunct
    Merge of 2 entries
      Merge block entry 1
      Conjunct        Index only retrieval of relation KSU
        Index name  KSU_IDX [0:0]
      Merge block entry 2
      Cross block of 3 entries
        Cross block entry 1
          Conjunct
          Merge of 1 entries
            Merge block entry 1
            Reduce  Conjunct
            Index only retrieval of relation KSU
              Index name  KSU_IDX [1:1]
        Cross block entry 2
          Conjunct
          Merge of 1 entries
            Merge block entry 1
            Index only retrieval of relation KOSEIUNIT_PSIYO
              Index name  KSU_IDX_1 [4:4]
        Cross block entry 3
          Conjunct        Aggregate-F1
          Index only retrieval of relation KSU
            Index name  KSU_IDX [6:6]
```

The view KSU_V contains a UNION All clause.

```
create view KSU_V
    (PROD, ATTCD, UGRP, UBLK, USEQ, PSIYO) AS
    select C2.PROD, C2.ATTCD, C2.UGRP, C2.UBLK, C2.USEQ, C2.PSIYO
        from KSU C2
    union all
    select C3.F1, C3.F2, C3.F3, C3.F4, C3.F5, C6.PSIYO
        from
            (select C5.PROD, C5.ATTCD, C5.UGRP, C5.UBLK, C5.USEQ
                from KSU C5
                group by C5.PROD, C5.ATTCD, C5.UGRP, C5.UBLK, C5.USEQ)
            as C3 (F1, F2, F3, F4, F5),
            (select C7.PROD, C7.UGRP, C7.UBLK, C7.USEQ, C7.PSIYO
                from KOSEIUNIT_PSIYO C7
                where ((((C7.PROD = C3.F1)
                    and (C7.UGRP = C3.F3))
                    and (C7.UBLK = C3.F4))
                    and (C7.USEQ = C3.F5)))
            as C6 (PROD, UGRP, UBLK, USEQ, PSIYO)
        where (not exists (select * from KSU C8
            where ((((((C8.PROD = C3.F1)
                and (C8.ATTCD = C3.F2))
                and (C8.UGRP = C3.F3))
                and (C8.UBLK = C3.F4))
                and (C8.USEQ = C3.F5))
                and (C8.PSIYO = C6.PSIYO))));
```

The problem also exists in both Rdb Release 7.1.5.1 as well as Rdb Release 7.2.1.3. However, the query performs really fast on Rdb Release 7.0.9.

There is no workaround for this problem since this is a regression caused by the fixes for Bugs 5567495 and 4747999 in Rdb Release 7.1.5.1.

The presence of the first leg of the UNION in the view causes the strategy to use the full index scan of the following offending index.

```
  Cross block entry 2
    Conjunct        Aggregate-F1    Conjunct
    Merge of 2 entries
      Merge block entry 1
      Conjunct        Index only retrieval of relation KSU
        Index name  KSU_IDX [0:0]                <==
      Merge block entry 2
```

The index KSU_IDX is defined as:

```
create index KSU_IDX on KSYU
        (PROD, ATTCD, UGRP, UBLK, USEQ, PSIYO);
```

This problem has been corrected in Oracle Rdb Release 7.1.5.2.

2.2 SQL Errors Fixed

2.2.1 SUBSTRING Truncation for View Defined in Program

Bug 6117796

A precompiled SQL or SQL Module language program which defined a view containing a column defined by using a SUBSTRING built−in function would receive truncated results in some cases when using the view for retrievals during the same program execution.

For example, suppose a Precompiled SQL program contained the following SQL statements:

```
EXEC SQL CREATE TABLE MOREGRUB (C1 VARCHAR (10), ID INT);
EXEC SQL CREATE VIEW X4 (S1, ID) AS
  SELECT (C1 FROM 2 FOR 4), ID
  FROM MOREGRUB;
EXEC SQL INSERT INTO MOREGRUB VALUES ('Pretzels', 1);
EXEC SQL SELECT S1 INTO :ch1 FROM X4 WHERE ID = 1;
```

After the above code was executed, the value in the host variable ":ch1" would be "re " instead of the correct value of "retz".

The problem can be worked around by defining the view external to the program or by selecting directly against the table instead of using a view.

This problem has been corrected in Oracle Rdb Release 7.1.5.2.

2.2.2 Unexpected PORT_LEN Error When Calling Storage Mapping Function

Bug 6129632

In prior releases, Oracle Rdb would create a special storage map function using the same name as the table's storage map. This function can be passed data values and have the partition number returned to the caller. In some cases, such calls will fail with an RDB−F−PORT_LEN error as shown in this example.

```
SQL> SELECT sample_map (acct_num) FROM sample;
%RDB-F-PORT_LEN, buffer length 25 does not match BLR description 29
```

This error occurs when the total length of the user data is a multiple of 4. In this case, the column ACCT_NUM is of type INTEGER (that is 4 octets in length). There is no workaround for this problem.

Once this release of Oracle Rdb has been installed, the following command can be used to regenerate a working function definition:

```
SQL> ALTER STORAGE MAP sample_map COMPILE;
SQL> COMMIT;
```

This command should be applied to any storage map that exhibits this error.

This problem has been corrected in Oracle Rdb Release 7.1.5.2. Oracle Rdb now correctly generates the storage mapping function.

2.2.3 Unexpected Failure of ALTER DATABASE to Enable PERSONA Support

Bug 6132645

In some cases, an ALTER DATABASE ... SECURITY CHECKING IS EXTERNAL (PERSONA IS ENABLED) would fail to enable PERSONA on the database. This happens in rare cases when the database has a very low number of buffers. In the reported case, the database was defined with NUMBER OF BUFFERS 10.

The workaround is to increase the number of buffers for the database to 100 or so to allow the buffer containing the RDB$DATABASE row to remain in memory.

This problem has been corrected in Oracle Rdb Release 7.1.5.2. This release of Rdb ensures that the RDB$DATABASE row is current during the update of the RDB$FLAGS column.

2.2.4 Unexpected Column Reordering After an ALTER TABLE ... ADD COLUMN Statement

Bug 6148536

In prior releases of Oracle Rdb, it was possible for SHOW TABLE on tables that were often altered to show that the columns had been unexpectedly reordered. This unusual case happens when the RDB$FIELD_ID values exceed 65535. Oracle discourages such tables in production as the very large RDB$FIELD_ID causes the null bit vector of these rows to be quite large. The many ALTER TABLE operations will also propagate description rows in the RDB$FIELD_VERSIONS table that will affect metadata loading by the Rdb Server.

The large RDB$FIELD_ID value will force the row to be followed by a null bit vector large enough to hold all fields. If compression is disabled, then this will lead to fragmented rows and excessive I/O during table processing. Therefore, the correct solution for this problem is to unload the table data and recreate the table. A SQL EXPORT and IMPORT would also have the same result.

The cause of the unexpected reordering was an assumption that the RDB$FIELD_ID would remain 16 bits in size and so the high order 16 bits of RDB$FIELD_POSITION were used to encode the AFTER COLUMN and BEFORE COLUMN clauses for ALTER TABLE. The overflow was incorrectly interpreted as a column reordering command.

This problem has been corrected in Oracle Rdb Release 7.1.5.2. Oracle Rdb now manages the reordering in a different way that no longer uses the RDB$FIELD_ID column values.

2.2.5 SET ALL CONSTRAINTS Causes SQLCODE −1005 With Compound Statement

Bugs 462035 and 1403770

If a Precompiled SQL or SQL Module Language program executed a statement with no active transaction while constraints were being evaluated immediately, the statement would fail with a SQLCODE of −1005 and associated Rdb error message "%RDB−E−BAD_TRANS_HANDL". Typically, the SQL statement might be a compound statement or a call to a stored or external procedure since these statements do not require a transaction to be active.

For example, the following Precompiled SQL program commits its transaction and then immediately executes a SQL compound statement. That compound statement would fail with a SQLCODE of −1005.

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
EXEC SQL DECLARE ALIAS FILENAME PERSONNEL;
EXEC SQL INCLUDE SQLCA;
main()
{
    EXEC SQL set transaction;
    printf("SET TRANS SQLCODE is %ld\n", SQLCA.SQLCODE);
    EXEC SQL set all constraints on;
    printf("SET ALL CONSTR ON SQLCODE is %ld\n", SQLCA.SQLCODE);
    EXEC SQL commit;
    printf("COMMIT SQLCODE is %ld\n", SQLCA.SQLCODE);
    EXEC SQL begin set transaction; end;
    printf("compound SQLCODE is %ld\n", SQLCA.SQLCODE);
}
```

This problem has been corrected in Oracle Rdb Release 7.1.5.2.

2.2.6 GET DIAGNOSTICS in Distributed Transaction Makes Transaction Non−distributed

Bug 972038

Under some circumstances, a compound statement containing a GET DIAGNOSTICS statement would cause a Precompiled SQL or SQL Module Language program built with "TRANSACTION_DEFAULT=DISTRIBUTED" to start a non−distributed transaction. The compound statement also needed a statement that requires a transaction context, such as an INSERT. In such a case, if there was no distributed transaction active, the SQL$PRE or SQL$MOD program would not start one prior to executing the compound statement. This would cause the server to implicitly start a non−distributed transaction for the database attach.

For example, if the following Precompiled C code were program compiled with a SQLOPTION of "TRANSACTION_DEFAULT=DISTRIBUTED", the condition could occur.

```
...
EXEC SQL DECLARE ALIAS FILENAME PERSONNEL;
...
main ()
{
    long rc;
    long status;
    short iosb[4];
    long flag = 2;
    long tid[4];
...
```

```
    status = sys$start_transw(
        0, /* efn */
        flag, /* flags */
        iosb, /* iosb */
        0, /* astadr */
        0, /* astprm */
        tid /* tid */
    );
...
    EXEC SQL BEGIN
        update employees set last_name = 'Toliver'
            where employee_id = '00164';
        get diagnostics :rc = row_count;
    END;
    printf("row count = %d\n", rc);
...
    status = sys$end_transw(
        0, /* efn */
        0, /* flag */
        iosb, /* iosb */
        0, /* astadr */
        0, /* astprm */
        tid /* tid */
    );
...
}
```

In the above example, the presence of the "GET DIAGNOSTICS" statement in the compound statement would cause the client program to not automatically join the distributed transaction initiated by the call to SYS$START_TRANSW(). Since there would then be no Rdb transaction in progress for the PERSONNEL database, the Rdb server would implicitly start a one−phase transaction independent from the application's distributed transaction. The call to SYS$END_TRANSW() would not then commit the Rdb transaction because it was not in a branch participating in the distributed transaction.

SQL$PRE and SQL$MOD have been modified so that when "TRANSACTION_DEFAULT=DISTRIBUTED" is specified, code is generated which does implicitly join any existing distributed transaction prior to executing a compound statement containing a "GET DIAGNOSTICS" statement.

The problem could be worked around by making sure a distributed transaction was active when such a compound statement was executed.

This problem has been corrected in Oracle Rdb Release 7.1.5.2.

2.2.7 IMPORT DATABASE Command Losing Some Command Line Database Attributes

Bug 6311200

In prior releases of Oracle Rdb, the SQL IMPORT DATABASE command would not correctly inherit the SHARED MEMORY IS PROCESS RESIDENT clause or the LARGE MEMORY IS ENABLED attribute of the GLOBAL BUFFERS ARE ENABLED clause. The corresponding attributes were always inherited from the interchange file (.RBR).

This has been corrected in Oracle Rdb Release 7.1.5.2. However, a workaround is to use an ALTER DATABASE statement immediately following the IMPORT DATABASE command to set these attributes. The RMU/EXTRACT/ITEM=DATABASE command can be used to verify these settings.

2.2.8 WAIT and NOWAIT Flags Not Used by Sequences

Bug 6487720

Prior versions of Oracle Rdb ingored the WAIT and NOWAIT flags defined for sequences. Therefore, all wait operations were controlled by the settings of the current transaction. When a transaction specified the NOWAIT clause, a reference to NEXTVAL might fail with a LOCK_CONFLICT error.

```
%RDB-E-LOCK_CONFLICT, request failed due to locked resource
-RDMS-F-LCKCNFLCT, lock conflict on client '........SEQ_....'
000000015F514553000000010000019000000
```

This problem has been corrected in Oracle Rdb Release 7.1.5.2. WAIT and NOWAIT clauses of the ALTER SEQUENCE and CREATE SEQUENCE command now function as documented in the Oracle Rdb SQL Reference Manual.

2.3 RDO and RDML Errors Fixed

2.3.1 %COBOL−F−AMBIGSYM, Ambiguous Reference With RDBPRE and COBOL

Bug 486335

Under certain circumstances, compiling an RDBPRE/COBOL program would result in generated code with duplicate names in a generated record definition. When the resulting COBOL code was submitted to the COBOL compiler, the following error message would be generated at the point the field with the duplicate name was used:

```
%COBOL−F−AMBIGSYM, Ambiguous reference − check name qualification ...
```

The record structures with the duplicate names were used by RDBPRE to store intermediate results during a cross−database update. The problem would manifest itself when columns in the source database had different types and/or sizes than the corresponding target columns in the other database. For example, consider a pair of Rdb databases defined by the following SQL statements.

```
create database filename essai1
create table a_table (
    a_zone1     SMALLINT,
    a_zone2     SMALLINT,
    a_zone3     INTEGER (2));
disconnect all;
create database filename essai
create table b_table (
    b_zone1     SMALLINT,
    b_zone2     INTEGER (3),
    b_zone3     SMALLINT (1));
disconnect all;
```

The generated code from the following RDBPRE/COBOL program would contain a record definition with a duplicate name.

```
IDENTIFICATION DIVISION.
PROGRAM−ID. tst.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING−STORAGE SECTION.
*
&RDB& INVOKE DATABASE b  = FILENAME "essai"
&RDB& INVOKE DATABASE a  = FILENAME "essai1"
*
PROCEDURE DIVISION.
DEB−PROG.
&RDB&    START−TRANSACTION ON B  USING (READ−WRITE
&RDB&                      RESERVING b.b_table            FOR SHARED READ)
&RDB&                    AND ON A  USING (READ−WRITE
&RDB&                      RESERVING a.a_table            FOR SHARED WRITE)
&RDB&    FOR   EC  IN b.b_table
&RDB&           WITH  EC.b_zone1  = 0
&RDB&           STORE ECC IN a.a_table USING
&RDB&                   ECC.a_zone2  = EC.b_zone2;
```

```
&RDB&                   ECC.a_zone3  = EC.b_zone3;
&RDB&           END_STORE
&RDB&    END_FOR
&RDB&    COMMIT
          stop run.
```

The problem could often be worked around by rearranging the column order in the STORE statement.

This problem has been corrected in Oracle Rdb Release 7.1.5.2.

2.3.2 %RDB−F−ARITH_EXCEPT With RDBPRE/COBOL and Literal in Expression

Bug 560785

In some cases, a STORE or MODIFY that involved an arithmetic expression with a literal in it would cause RDBPRE/COBOL to generate bad code. Specifically, the generated code would have a type mismatch between a BLR message and the corresponding items in the generated COBOL record for that message. At run time, this would often lead to error messages such as: "%RDB−F−ARITH_EXCEPT" and/or "−COSI−F−INTOVF".

The following example of a program that demonstrates the problem uses a PERSONNEL database with the JOBS relation altered so that the MAXIMUM_SALARY and MINIMUM_SALARY fields have a type of "signed word scale 0". The JOBS.RCO program from SQL$SAMPLES is modified as described below.

- The fields WS−SAL−MIN and WS−SAL−MAX are modified to be four byte integers as follows:

  ```
  01 WS-SAL-MIN                   PIC S9(4) COMP.
  01 WS-SAL-MAX                   PIC S9(4) COMP.
  ```
- The STORE−JOBS subroutine is modified to set the minimum and maximum salary variables to "1" and use an arithmetic expression involving a literal in the RDO statement for storing the jobs records as follows:

  ```
  STORE-JOBS.
          MOVE 1 TO WS-SAL-MIN.
          MOVE 1 TO WS-SAL-MAX.
  &RDB&   STORE J IN JOBS
  &RDB&      USING
  &RDB&           J.JOB_CODE              = J-CODE;
  &RDB&           J.WAGE_CLASS            = W-CLASS;
  &RDB&           J.JOB_TITLE             = J-TITLE;
  &RDB&           J.MINIMUM_SALARY        = WS-SAL-MIN + 1;
  &RDB&           J.MAXIMUM_SALARY        = WS-SAL-MAX + 2;
  &RDB&    END_STORE
  ```

When the modified JOBS.RCO program was compiled and linked, the resulting code would have a mismatch between the minimum and maximum salary fields in the generated COBOL record versus the message in the BLR. This caused run time errors.

This problem has been corrected in Oracle Rdb Release 7.1.5.2.

2.4 RMU Errors Fixed

2.4.1 RMU/VERIFY %RMU−I−BTRDUPCAR Diagnostic Message Changed To %RMU−E−BTRDUPCAR

The %RMU−I−BTRDUPCAR diagnostic message that could be displayed when verifying Oracle Rdb database Sorted Ranked indexes indicates that the index structure is corrupt and that the index needs to be rebuilt since the cardinality specified in an index entry is inconsistent with the cardinality computed from the duplicate list for the entry. This will cause wrong results for SQL queries like COUNT(*). Therefore, the RMU−I−BTRDUPCAR diagnostic message severity level has been changed from Informational to Error so that this message will not be ignored.

The following example shows the %RMU−I−BTRDUPCAR diagnostic message that was put out when RMU/VERIFY detected a corruption in the structure of a Sorted Ranked index in an Rdb database. This indicated that the index needed to be rebuilt.

```
$ RMU/VERIFY/INDEX=bad_index mf_personnel
%RMU−I−BTRDUPCAR, Inconsistent duplicate cardinality (C1) of 16777318  specified
                  for entry 1 at dbkey 58:10:0.
                  Actual count of duplicates is 16777317.
%RMU−I−BTRROODBK, root dbkey of B−tree is 58:10:0
```

The following example shows the %RMU−E−BTRDUPCAR diagnostic message which will now be output with an "Error" instead of an "Informational" severity to make sure that the index corruption is brought to the user's attention.

```
$ RMU/VERIFY/INDEX=bad_index mf_personnel
%RMU−E−BTRDUPCAR, Inconsistent duplicate cardinality (C1) of 16777318  specified
                  for entry 1 at dbkey 58:10:0.
                  Actual count of duplicates is 16777317.
%RMU−I−BTRROODBK, root dbkey of B−tree is 58:10:0
```

This problem has been corrected in Oracle Rdb Release 7.1.5.2.

2.4.2 RMU/REPAIR to Move Snapshot File Can Fail With Bad New File Specification

Bug 4083632

A concealed device name in the default file specification for a snapshot file and specifying a new snapshot location can cause the RMU/REPAIR to fail. See the following example.

```
$ RMU/REPAIR/INITIALIZE=(SNAPSHOT=CONFIRM)/AREA=MYDB_DATA −
    $1$DKA100:[USER.][SUB]MYDB
Area MYDB_DATA snapshot filename [$1$DKA100:[USER.][SUB]MYDB_DATA.SNP;1]:
    >>> User response: $1$DKA100:[USER]MYDB_DATA.SNP
Area MYDB_DATA snapshot file allocation [100]:
%RMU−F−FILACCERR, error creating database file $1$DKA100:[USER.]
[USER]MYDB_DATA.SNP;1
−RMS−E−DNF, directory not found
```

This problem has been corrected in Oracle Rdb Release 7.1.5.2.

2.4.3 RMU−F−FILACCERR, Error Truncating File on RMU/BACKUP/AFTER

Bug 5629652

An RMU/BACKUP/AFTER of a single extensible after image journal file that had extended and then had been disabled prior to the backup of the journal file failed with the following error:

```
%RMU−F−FILACCERR, error truncating file
 −SYSTEM−W−ACCONFLICT, file access conflict
```

The following example shows the RMU−F−FILACCERR which occurred during the RMU/BACKUP/AFTER of a single extensible after image journal file that had extended and had then been disabled prior to the backup of the journal file.

```
$ SQL
create data file foo;
create table tbl1 (c1 char(50),c2 char(50));
commit;
exit
$ rmu/backup foo foo
$ rmu/set after/enable/add=(name=aij1,file=aij1) foo
$ SQL
att 'f foo';
@load.sql
exit
$ rmu/set after/disable foo
$ rmu/backup/after/log foo fooaij
   %RMU−F−FILACCERR, error truncating file
   −SYSTEM−W−ACCONFLICT, file access conflict
```

A workaround for this problem is not to disable the journal file prior to the backup of the journal file.

This problem has been corrected in Oracle Rdb Release 7.1.5.2.

2.4.4 RMU/COLLECT/INDEXES Gave No Error If An Index Did Not Exist

Bug 6316729

If RMU/COLLECT OPTIMIZER_STATISTICS /INDEX=(A,B,C) or RMU/SHOW OPTIMIZER_STATISTICS /INDEX=(A,B,C) was given the name of an index that did not exist in an Oracle Rdb database, it ignored the index but continued and returned no error.

This problem has been fixed and now if the user specifies a named index that does not exist in the database, the RMU/COLLECT OPTIMIZER or RMU/SHOW OPTIMIZER command will be aborted and an error will be returned.

```
$ rmu/collect opt mf_personnel /notables /log /index=bad_name
Start loading tables... at 28−AUG−2007 16:16:14.54
```

```
Done loading tables.... at 28-AUG-2007 16:16:14.54
Start loading indexes... at 28-AUG-2007 16:16:14.54
%RMU-F-INDNOTFND, index BAD_NAME does not exist in this database
%RMU-F-FTL_COL_STAT, Fatal error for COLLECT OPTIMIZER_STATISTICS operation at
28-AUG-2007 16:16:14.54
```

The following example shows the problem that has been fixed. Even though the named index "bad_name" does not exist, RMU/COLLECT OPTIMIZER_STATISTICS gives no error but continues and ignores the index.

```
$ rmu/collect opt mf_personnel /notables /log /index=bad_name
Start loading tables... at 28-AUG-2007 16:14:21.04
Done loading tables.... at 28-AUG-2007 16:14:21.05
Start loading indexes... at 28-AUG-2007 16:14:21.05
Done loading indexes.... at 28-AUG-2007 16:14:21.06
Start collecting btree index stats... at 28-AUG-2007 16:14:21.12
Done collecting btree index stats.... at 28-AUG-2007 16:14:21.12
Start collecting table & hash index stats... at 28-AUG-2007 16:14:21.12
Done collecting table & hash index stats.... at 28-AUG-2007 16:14:21.12
Start collecting workload stats... at 28-AUG-2007 16:14:21.17
Maximum memory required (bytes) = 0
Done collecting workload stats.... at 28-AUG-2007 16:14:21.18
Start calculating stats... at 28-AUG-2007 16:14:21.18
Done calculating stats.... at 28-AUG-2007 16:14:21.18
Start writing stats... at 28-AUG-2007 16:14:21.21
Done writing stats.... at 28-AUG-2007 16:14:21.21
```

This problem has been corrected in Oracle Rdb Release 7.1.5.2.

2.4.5 RMU OPTIMIZER_STATISTICS Commands Gave Wrong Fatal Error Exit Message

RMU/COLLECT OPTIMIZER_STATISTICS, RMU/DELETE OPTIMIZER_STATISTICS and RMU/INSERT OPTIMIZER_STATISTICS returned a fatal error exit message which did not name correctly the operation being performed. Note that the error was handled correctly but the fatal message put out before Oracle Rdb RMU exited was incorrect.

This problem has been fixed and now a fatal error exit message appropriate to the specific command operation will be output.

```
$ rmu/collect optimizer_statistics baddb
%RMU-W-BADDBNAME, can't find database root
DEVICE:[DIRECTORY]BADDB.RDB;
-RMS-E-FNF, file not found
%RMU-F-CANTOPNROO, cannot open root file "BADDB"
%RMU-F-FTL_COL_STAT, Fatal error for COLLECT OPTIMIZER_STATISTICS operation at
 29-AUG-2007 15:44:07.54
$ rmu/delete optimizer_statistics baddb
%RMU-W-BADDBNAME, can't find database root
DEVICE:[DIRECTORY]BADDB.RDB;
-RMS-E-FNF, file not found
%RMU-F-CANTOPNROO, cannot open root file "BADDB"
%RMU-F-FTL_DEL_STAT, Fatal error for DELETE OPTIMIZER_STATISTICS operation at
 29-AUG-2007 15:44:30.18
$ rmu/insert optimizer_statistics baddb
%RMU-W-BADDBNAME, can't find database root
DEVICE:[DIRECTORY]BADDB.RDB;
```

```
-RMS-E-FNF, file not found
%RMU-F-CANTOPNROO, cannot open root file "BADDB"
%RMU-F-FTL_INS_STAT, Fatal error for INSERT OPTIMIZER_STATISTICS operation at
 29-AUG-2007 15:44:50.48
```

The following example shows the problem that has been fixed. The fatal error exit message %RMU-F-FTL_ANA was returned by RMU. It did not correctly describe the command operation being performed.

```
$ rmu/collect optimizer_statistics baddb
%RMU-W-BADDBNAME, can't find database root
DEVICE:[DIRECTORY]BADDB.RDB;
-RMS-E-FNF, file not found
%RMU-F-CANTOPNROO, cannot open root file "BADDB"
%RMU-F-FTL_ANA, Fatal error for ANALYZE operation at 29-AUG-2007
 15:33:52.70
$ rmu/delete optimizer_statistics baddb
%RMU-W-BADDBNAME, can't find database root
DEVICE:[DIRECTORY]BADDB.RDB;
-RMS-E-FNF, file not found
%RMU-F-CANTOPNROO, cannot open root file "BADDB"
%RMU-F-FTL_ANA, Fatal error for ANALYZE operation at 29-AUG-2007
 15:34:05.65
$ rmu/insert optimizer_statistics baddb
%RMU-W-BADDBNAME, can't find database root
DEVICE:[DIRECTORY]BADDB.RDB;
-RMS-E-FNF, file not found
%RMU-F-CANTOPNROO, cannot open root file "BADDB"
%RMU-F-FTL_ANA, Fatal error for ANALYZE operation at 29-AUG-2007
 15:34:19.03
```

This problem has been corrected in Oracle Rdb Release 7.1.5.2.

2.4.6 RMU/VERIFY/LOG VMS Exit Status Could Be Incorrect

Big 6315971

If RMU/VERIFY/LOG was specified and no errors or warnings occurred when an Oracle Rdb database was verified, the VMS exit status would be incorrectly set to a log message with Informational severity instead of a success status. This did not occur if /LOG was not specified. This problem has been fixed and now a success status will be output if /LOG is specified as it already is if /LOG is not specified.

The following example shows the problem that has been fixed. A successful RMU/VERIFY of an Rdb database would show a successful VMS exit status if /LOG was not specified but if /LOG was specified the exit status would be that of an informational log message. Now in both cases a successful VMS exit status will be set.

```
$ rmu/verify/all mf_personnel
$ show symbol $status
  $STATUS == "%X10000001"
$ rmu/verify/all/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
 %RMU-I-ENDVCONST, completed verification of constraints for database
  DEVICE:[DIRECTORY]MF_PERSONNEL.RDB;1
```

```
 %RMU-S-ENDVERIFY, elapsed time for verification :    0 00:00:02.22
$ show symbol $status
  $STATUS == "%X12C8B51B"
```

This problem has been corrected in Oracle Rdb Release 7.1.5.2.

Chapter 3
Software Errors Fixed in Oracle Rdb Release 7.1.5.1

This chapter describes software errors that are fixed by Oracle Rdb Release 7.1.5.1.

3.1 Software Errors Fixed That Apply to All Interfaces

3.1.1 %RDB−E−REQ_NO_TRANS When Fetching from a Cursor

Bug 3000645

Under some circumstances, SQL would generate a %RDB−E−REQ_NO_TRANS error after the execution of a SQL stored procedure or multi−statement procedure which left the process with no currently active transaction. This error was often encountered in conjunction with using a hold cursor.

The following SQL creates a stored procedure (TX_END) which, if executed, would set up the state where the problem would have been encountered. Note that the stored procedure simply ends any active transaction and returns.

```
create module tx_end
    language sql
    declare transaction read only

procedure tx_end();
begin
declare :v_active tinyint  default 0;
get diagnostics :v_active = TRANSACTION_ACTIVE;
if (:v_active <> 0) then
    rollback;
end if;
end;
end module;
```

The problem could be worked around by ending the transaction with an explicit SQL statement in lieu of doing it inside a stored procedure.

This problem has been corrected in Oracle Rdb Release 7.1.5.1.

3.1.2 Distinct Query Bugchecks With Bitmap Scan Enabled

Bug 5295755

During the second execution of a "Distinct" query, with Bitmap Scan enabled, a bugcheck would occur.

```
set flags 'nobitmap,strategy,detail';
select  distinct test_nbr from t1 where asn='10000';
Tables:
  0 = T1
Reduce: 0.TEST_NBR
Leaf#01 Sorted 0:T1 Card=141
  Bool: 0.ASN = '10000'
  FgrNdx  IND_TN [0:0] Fan=17
  BgrNdx1 IND_ASN [1:1] Fan=14
    Keys: 0.ASN = '10000'
```

```
     TEST_NBR
        1.000
        2.000
2 rows selected

set flags 'bitmap,strategy,detail';
select  distinct test_nbr from t1 where asn='10000';
%RDMS-I-BUGCHKDMP, generating bugcheck dump file RDSBUGCHK.DMP;
```

There were two exceptions possible:

```
***** Exception at 01001154 : RDMS$$EXE_CLOSE + 00000574
%COSI-F-BUGCHECK, internal consistency failure
```

or,

```
***** Exception at 0124631C : KOD$ROLLBACK + 0000032C
%COSI-F-BUGCHECK, internal consistency failure
```

The query works if the 'bitmap' is NOT enabled at the second execution, as in the following example.

```
set flags 'nobitmap,strategy';
select  distinct test_nbr from t1 where asn='10000';
Tables:
  0 = T1
Reduce: 0.TEST_NBR
Leaf#01 Sorted 0:T1 Card=141
  Bool: 0.ASN = '10000'
  FgrNdx  IND_TN [0:0] Fan=17
  BgrNdx1 IND_ASN [1:1] Fan=14
    Keys: 0.ASN = '10000'
     TEST_NBR
        1.000
        2.000
2 rows selected
```

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

1. The main select is a distinct query from a table with a filter predicate.
2. The table has two indices, each index contains one segment column.
3. The index that contains the column referenced by the filter predicate is applied as the background index.
4. The index that contains the column referenced by the distinct function is applied as the foreground index.
5. The SQL flag 'Bitmap' does not need to be enabled at the first execution, but it must be explicitly turned on before the second execution.
6. The strategy dynamic tactic must be "Sorted".
7. The background index scan completes sucessfully.

This problem affects the following Oracle Rdb versions: V7.1.4.4, V7.1.4.5, V7.1.5, V7.2.0.1, V7.2.0.2 and V7.2.1. This problem has been corrected in Oracle Rdb Release 7.1.5.1.

3.1.3 Program Fails With SYSTEM−F−ACCVIO in RDB$SHARE71

Bugs 5637359 and 5654859

Application programs which use distributed transactions initiated by the application (typically by a call to SYS$START_TRANS) may fail with Rdb V7.1.4.5.0 or V7.1.4.5.1 with a %SYSTEM−F−ACCVIO. The stack dump will contain an entry similar to the following:

```
%TRACE−F−TRACEBACK, symbolic stack dump follows
  image    module    routine             line      rel PC           abs PC
 RDB$SHARE71                                 0 00000000000CCF68 00000000021ECF68
```

The "abs PC" address will vary but the "rel PC" value in the above example is a characteristic signature of this problem.

The problem involves a read access to application memory and has no potential for corruption of application memory. Since it occurs during the starting of a transaction, the potential for application data loss is minimal and there is no chance of loss of transactional integrity. There is, however, no way for the application process to recover from the error and continue.

This problem was corrected in Oracle Rdb Release 7.1.5 but the Release Note was mistakenly left out.

3.1.4 Unexpected Query Failure With RDB−E−NO_RECORD Error

Bug 5846989

In prior releases of Oracle Rdb, it was possible for queries that used the MIN or MAX function to fail with the following error.

```
%RDB−E−NO_RECORD, access by dbkey failed because dbkey is
no longer associated with a record
−RDMS−F−NODBK, −524:22806:1 does not point to a data record
```

The DBKEY being displayed is an encoded DBKEY referencing the duplicate chain for the matching key.

This problem occurs under the following conditions:

- The query is solved using the Dynamic optimizer. This often means that Rdb has a choice of indices which could be used to solve the query.
- The selected index allows duplicate values.
- The minimum or maximum value contained more than one matching row.

As a workaround, the dynamic optimizer can be disabled using the logical name RDMS$SET_FLAGS or the SET FLAGS command with the "MAX_STABILITY" keyword, or by defining the logical name RDMS$MAX_STABILITY to the value 1.

An alternate workaround is to start the transaction using the clause ISOLATION LEVEL REPEATABLE READ.

This problem has been corrected in Oracle Rdb Release 7.1.5.1.

3.1.5 AIJ Backup Operation Aborts With NONAME−F−NOMSG Message Number 00000004

Bug 5890612

In rare cases, an after−image journal backup operation may fail with an unexpected incorrect status value. The actual value may vary, but at least one customer report of the problem indicated a value of 00000004. A bugcheck dump file "footprint" of this problem is:

```
***** Exception at 0054D94C : AIJBCK$GET_NEXT_JOURNAL + 00000CFC
Saved PC = 005452E8 : AIJBCK$FULL_BACKUP + 00000FF8
Saved PC = 00543C0C : AIJBCK$BACKUP + 0000113C
Saved PC = 0040E7A4 : RMUCLI$BACKUP_AIJ + 00000904
Saved PC = 003A2414 : RMU_DISPATCH + 00000484
Saved PC = 003A1AE8 : RMU_STARTUP + 000004E8
Saved PC = 001E002C : RMU$MAIN + 0000002C
```

This problem has been corrected in Oracle Rdb Release 7.1.5.1. The errant status value was the result of an uninitialized return status being passed back. The correct status is now returned.

3.1.6 Adding a Large AIJ File to a DB Fails With Either an ACCVIO or OPCDEC Error

Bug 5852864

Creating an after image journal file for a database fails with either an ACCVIO or an OPCDEC error. See the following example.

```
SQL> CREATE DATABASE
     ...
     PRESTARTED TRANSACTIONS ARE on
        (WAIT 1 MINUTES FOR TIMEOUT)
     ...
SQL> ALTER DATABASE FILENAME myDB ADD JOURNAL journal01 FILENAME 'JOURNAL01'
ALLOCATION IS 7000000 BLOCKS;
%SYSTEM−F−ACCVIO, access violation, reason mask=00,
                  virtual address=000000000000000C, PC=000000000000000C,...
```

This was caused by a bug in the code which prematurely cleared a synchronization flag. This allowed the request created by the expired prestarted transactions timer to execute before the current request had completed. The side effect of this was a stack corruption.

As a workaround, use a larger prestarted transactions timer value or disable the prestarted transactions timer completely using the SQL ALTER DATABASE statement.

This problem has been corrected in Oracle Rdb Release 7.1.5.1.

3.1.7 Simple Query Fails with ARITH_EXCEPT

Bug 5941200

A simple query fails with an arithmetic exception, as in the example below.

```
SELECT * FROM auth_control WHERE scheme_code = 'aaa'
AND task_ind = 'x'  AND in_use_ind = 'g';
%RDB-E-ARITH_EXCEPT, truncation of a numeric value at runtime
-SYSTEM-F-HPARITH, high performance arithmetic trap, Imask=00000000,
Fmask=00000400, summary=04, PC=00000000008BADFC, PS=0000001B
-SYSTEM-F-FLTDIV, arithmetic trap, floating/decimal divide by zero at
PC=00000000008BADFC, PS=0000001B
```

The query works if the SQL flag 'index_column_group' is disabled, as in the following example.

```
set flags 'noindex_column_group';
set flags 'stra,detail';

SELECT * FROM auth_control WHERE scheme_code = 'aaa'
AND task_ind = 'x'  AND in_use_ind = 'g';
0 rows selected
```

The query also works if RMU/COLLECT OPTIMIZER is applied, as in the following example.

```
$rmu/collect optimizer <database>
$SQL$
ATT 'FILE <database>';

SELECT * FROM auth_control WHERE scheme_code = 'aaa'
AND task_ind = 'x'  AND in_use_ind = 'g';
0 rows selected
```

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

1. The query selects from a table with three equality predicates, for example, COL1 = 1 AND COL2 = 2 AND COL3 = 3.
2. The table contains an index with all the above three columns in a contiguous order, for example, COL1, COL2, COL3, COL4.
3. One of the segment columns contains a zero value for the prefix cardinality. To find out this information, you need to set the SQL flags 'costing' and 'cursor_stat', and execute the query again. Here is the information on the prefix cardinality dumped out by the query:

   ```
   Segment group factors of 4 segments are estimated.
      COL1     GROUP_FACTOR  3.2407680E+00     PREFIX_CARD 1
      COL2     GROUP_FACTOR  2.4153826E+00     PREFIX_CARD 1
      COL3     GROUP_FACTOR  1.8002133E+00     PREFIX_CARD 0
      COL4     GROUP_FACTOR  1.3417203E+00     PREFIX_CARD 0
   ```

This problem affects the following Oracle Rdb releases: V7.1.4.4, V7.1.4.5, V7.1.5, V7.2.0.1, V7.2.0.2 and V7.2.1.0. This problem has been corrected in Oracle Rdb Release 7.1.5.1.

3.1.8 %MTH−F−SQUROONEG Error from RMU Workload Collect

Bug 4106407

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

```
%MTH−F−SQUROONEG, square root of negative value
  user PC 00000003
  %RMU−F−FATALOSI, Fatal error from the Operating System Interface.
  %RMU−F−FTL_ANA, Fatal error for ANALYZE operation at 19−DEC−2004 05:29:36.83
```

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

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

This problem has been corrected in Oracle Rdb Release 7.1.5.1.

3.1.9 ALTER DATABASE ... SHARED MEMORY IS PROCESS Did Not Disable RESIDENT

Previously it was not possible, while using SQL, to disable the RESIDENT attribute for database shared memory once it had been enabled.

In the following example, note that once enabled, the RESIDENT attribute does not get cleared when using the ALTER statement.

```
$ SQL$ CREATE DATA FILE FOO SHARED MEMORY IS PROCESS RESIDENT;
$ RMU/DUMP/HEAD/OUT=X.X FOO
$ SEARCH X.X MAPPED
    Database will be mapped in process space
      Shared memory will be mapped resident (OpenVMS Alpha only)
$ SQL$ ALTER DATA FILE FOO SHARED MEMORY IS PROCESS;
$ RMU/DUMP/HEAD/OUT=X.X FOO
$ SEARCH X.X MAPPED
    Database will be mapped in process space
      Shared memory will be mapped resident (OpenVMS Alpha only)
```

As a workaround, the "RMU/SET SHARED_MEMORY/TYPE=" command can be used to switch between SYSTEM, PROCESS and RESIDENT.

This problem has been corrected in Oracle Rdb Release 7.1.5.1. Using "ALTER DATABASE ... SHARED MEMORY IS PROCESS" now correctly clears the setting of the RESIDENT attribute.

3.1.10 Incomplete Error Handling for COSI−E−WORK_DEV

Bug 4898352

In the event that an Rdb sort work file is not a local, random access file, Rdb reported an incomplete error message. In the case that the database in question was remote, the remote RDBSERVER would fail with an access violation. For example, if the error were in an interactive SQL session for a local database, the incomplete error message would appear as follows:

```
%COSI-E-WORK_DEV, work file !AS must be on random access local device
```

In the above example, the "!AS" should have been replaced by the name of the problem sort work file, for example "NL:[]SORTWORK.TMP;".

If the error occurred on a remote database, the RDBSERVER would fail and the local session would see the following error messages:

```
%RDB-F-IO_ERROR, input or output error
-SYSTEM-F-LINKABORT, network partner aborted logical link
```

In the remote case, the NETSERVER.LOG for that session on the remote system would record a "%SYSTEM-F-ACCVIO" error.

This problem has been corrected in Oracle Rdb Release 7.1.5.1.

3.1.11 Wrong Result From Query With NOT NULL Test

Bug 6041167

A wrong result may be reported by a query with a NOT NULL test, as in the following example.

```
SHOW INDEX T1_NDX;
Indexes on table T1:
T1_NDX                          with column COL_DATA
                                and column KEY_ID
  Duplicates are allowed
  Type is Sorted Ranked
    Duplicates are Compressed Bitmaps
  Key suffix compression is DISABLED
  Node size  430

SEL COL_DATA,KEY_ID,NUMBER FROM T1;
Get     Retrieval by index of relation T1
  Index name  T1_NDX [0:0]
    COL_DATA        KEY_ID        NUMBER
          55          NULL      96507257
          66          NULL      96509951
          68          NULL      96508028
3 rows selected

SELECT COUNT(*) FROM T1 WHERE KEY_ID IS NOT NULL;
Aggregate       Conjunct        Index only retrieval of relation T1
  Index name  T1_NDX [0:0]       Index counts lookup

           3
1 row selected
```

The query works if the column of the NOT NULL predicate is defined as the leading segment, as in the following example.

```
DROP INDEX T1_NDX;
CREATE INDEX T1_NDX  ON T1 (KEY_ID,COL_DATA)
    TYPE IS SORTED RANKED;
SELECT COUNT(*) FROM T1 WHERE KEY_ID IS NOT NULL;
Aggregate       Index only retrieval of relation T1
  Index name  T1_NDX [0:0]       Index counts lookup

           0
1 row selected
```

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

1. The query selects the total count where the column is NOT null.
2. The column is defined as the trailing segment of the SORTED RANKED index.
3. The "Index counts lookup" was the chosen strategy.

A workaround for this problem is to disable the "Index counts lookup" for any affected queries. This is achieved using SET FLAGS 'NOCOUNT_SCAN'.

This problem affects the following Oracle Rdb releases: V7.1.4.4, V7.1.4.5, V7.1.5, V7.2.0.1, V7.2.0.2, V7.2.1, V7.2.1.1 and V7.2.1.2. This problem has been corrected in Oracle Rdb Release 7.1.5.1.

3.1.12 Possible Shared Memory Corruption When Multiple Databases Attached

Bug 5841667

Starting with Oracle Rdb Release 7.1.4 and Oracle Rdb Release 7.2, it was possible for shared memory to become corrupt. The corruption often would appear as (or would be caused by) data from one Rdb root file being written into the shared memory for another database. Once this corruption has occurred, reliability and functionality of the database and database users can be compromised.

Conditions leading to this corruption include:

- Processes accessing multiple databases
- Multiple database users
- Databases accessed from multiple nodes in a cluster
- Databases configured with "NUMBER OF CLUSTER NODES" ... "MULTIPLE INSTANCE"

The memory corruption was caused by incorrect IO buffer synchronization while refreshing root file information in to shared memory.

This problem has been corrected in Oracle Rdb Release 7.1.5.1. Oracle strongly recommends that customers with applications or procedures that may attach to more than one database at a time upgrade to Oracle Rdb Release 7.1.5.1 or later to avoid potential memory corruption problems.

3.1.13 Bugcheck at COSI$TIMER_GET_REQIDT With RDMS−F−NOREQIDT

Bug 6069358

Applications using the fast commit feature that periodically detach and reattach to a database within the same program may eventually run out of Rdb timer blocks and bugcheck with a footprint similar to the following.

```
***** Exception at 01235994 : COSI$TIMER_GET_REQIDT + 00000294
%RDMS-F-NOREQIDT, reached internal maximum number of simultaneous
timer requests
Saved PC = 01235C38 : COSI_TIMER_SET + 00000288
Saved PC = 01235DA8 : COSI_TIMER_SLEEP + 00000078
Saved PC = 012A7548 : KOD$COMMIT + 00000508
Saved PC = 01079148 : RDMS$$INT_COMMIT_TRANSACTION + 00000328
Saved PC = 01078D14 : RDMS$TOP_COMMIT_TRANSACTION + 000001E4
```

The following example script fragment demonstrates this failure using a fast commit checkpoint interval of 1 second and a delay between transactions of just over two seconds (to allow enough time for the checkpoint timer to fire twice):

```
    .
    .
    .
$ DEFINE SQL$DATABASE DKA0:[DB]DB.RDB
$ DEFINE RDM$BIND_CKPT_TIME 1
    .
    .
    .
$ SQL$
    INSERT INTO C1 VALUES (1);
    COMMIT;
    $ WAIT 0:0:2.02
    DISCONNECT ALL;
        .
        . Repeat the sequence 101 times
        .
    INSERT INTO C1 VALUES (1);
    COMMIT;
    $ WAIT 0:0:2.02
    DISCONNECT ALL;
%RDMS-I-BUGCHKDMP, generating bugcheck dump file
DISK$USER:[USER]RDSBUGCHK.DMP;
```

This problem has been corrected in Oracle Rdb Release 7.1.5.1. The problem was caused by an internal timer data structure being allocated but not being deallocated if the timer had expired. If the timer had not expired, the internal timer data structure was correctly deallocated. Thus, in some cases, the timer data structure was being "leaked" which could eventually lead to the bugcheck exception of "RDMS-F-NOREQIDT, reached internal maximum number of simultaneous timer requests".

3.2 SQL Errors Fixed

3.2.1 Some SQL Dialect−required Warnings Not Delivered

Bugs 3651847 and 4532451

The required warnings (information codes) for such things as rows eliminated for nulls (%RDB−I−ELIM_NULL) and string truncation (%RDB−I−TRUN_RTRV) were not being returned for singleton SELECT and singleton UPDATE statements; that is, statements that return a single row using the INTO clause. This could be demonstrated with a PERSONNEL database using the following interactive SQL commands:

```
SQL> set dialect 'sql92';
SQL> attach 'filename sql$database';
SQL>
SQL> ! Force a row to contain NULL for SALARY_AMOUNT
SQL> update salary_history
cont> set salary_amount = NULL
cont> where employee_id = '00471'
cont>   and salary_end = date vms'20−Aug−1981';
1 row updated
SQL>
SQL> declare :avg_sal integer(2);
SQL>
SQL> ! No informational generated (but is expected)
SQL> select avg(salary_amount) into :avg_sal
cont> from salary_history where employee_id = '00471'
cont>  and salary_end >= date vms'1−AUG−1970';
SQL> show sqlca
SQLCA:
        SQLCAID:        SQLCA           SQLCABC:        128
        SQLCODE:        0
        SQLERRD:        [0]: 0
                        [1]: 0
                        [2]: 1
                        [3]: 0
                        [4]: 0
                        [5]: 0
        SQLWARN0:       0       SQLWARN1:       0       SQLWARN2:       0
        SQLWARN3:       0       SQLWARN4:       0       SQLWARN5:       0
        SQLWARN6:       0       SQLWARN7:       0
        SQLSTATE:       00000
SQL> print :avg_sal;
        AVG_SAL
       60893.86
SQL> rollback;
```

The required SQLCODE and SQLSTATE values are now returned. In the example above, this causes the following differences:

```
SQL> select avg(salary_amount) into :avg_sal
cont> from salary_history where employee_id = '00471'
cont>  and salary_end >= date vms'1−AUG−1970';
%RDB−I−ELIM_NULL, null value eliminated in set function
SQL> show sqlca
SQLCA:
```

```
        SQLCAID:        SQLCA           SQLCABC:        128
        SQLCODE:        1003
        SQLERRD:        [0]: 0
                        [1]: 0
                        [2]: 1
                        [3]: 0
                        [4]: 0
                        [5]: 0
        SQLWARN0:       0       SQLWARN1:       0       SQLWARN2:       0
        SQLWARN3:       0       SQLWARN4:       0       SQLWARN5:       0
        SQLWARN6:       0       SQLWARN7:       0
        SQLSTATE:       01003
%RDB-I-ELIM_NULL, null value eliminated in set function
```

This problem has been corrected in Oracle Rdb Release 7.1.5.1.

3.2.2 Declared Variables Ignored by Oracle Dialect Dynamic Statements

Bug 5847260

In previous versions of Oracle Rdb, it was possible that dynamic statements would ignore variables created with the DECLARE syntax. This occurred when the dialect was set to ORACLE LEVEL1 or ORACLE LEVEL2.

This problem has been corrected in Oracle Rdb Release 7.1.5.1. The Oracle dialect now correctly identifies these as declared variables and not as parameter markers.

3.2.3 Unexpected RTN_FAIL/BAD_REQ_HANDLE Reported when Stored Function Called with All Constant Parameters

Bug 4764496

In prior releases of Oracle Rdb, it was sometimes possible for a query to fail with the error RTN_FAIL for "(unknown)" as the routine name, and a BAD_REQ_HANDLE secondary message.

The following shows an example using the REPLACE function from the SQL_FUNCTIONS routines.

```
select
  replace('hallo','a','e')
  from TAB1
 where COL2 = date'2006-06-29'
   and (  COL1 = 'AA'
           or
          COL1 = 'BB' );
%RDB-E-RTN_FAIL, routine "(unknown)" failed to compile or execute successfully
-RDB-E-BAD_REQ_HANDLE, invalid request handle
```

This problem occurred when the query optimizer tried to make use of the constant expression during query construction. Unfortunately, the referenced routine (in the example the function was named REPLACE) was not yet compiled for use by the query, and hence the name reported was unknown. Once the function is loaded, this query will succeed.

Some workarounds for this problem include:

- Define the function with the NOT DETERMINISTIC option to prevent it being considered a constant expression in this case.
- Execute ALTER FUNCTION ... COMPILE to pre−load the function prior to executing queries that reference it.
- Execute ALTER MODULE ... COMPILE to pre−load all functions in a module.
- Replace the constant expression with a varying column value.

This problem has been corrected in Oracle Rdb Release 7.1.5.1.

3.2.4 SYSTEM−F−ROPRAND With Large Command Line

Bug 4117738

With a very large command line, it was possible to receive a SYSTEM−F−ROPRAND message with interactive SQL. Furthermore, SQL was left in a state such that all subsequent commands resulted in a SYSTEM−F−ROPRAND error. In the example in the bug report, a command file consisting of a single line of over 11,000 bytes caused the problem to manifest itself.

The following sequence shows the problem symptom using a command file named "BIGCMD.SQL" which is too large to reproduce here.

```
SQL> @BIGCMD.SQL
%SYSTEM−F−ROPRAND, reserved operand fault at PC=0000000000F4DDA8, PS=0000001B
```

Note that a variety of other symptoms could also occur with such a large command line such as an access violation.

Note also that even after the problem in SQL is corrected, the command line in the example above contained a query that needed levels of recursion greater than the default size of Rdb's executive mode stack. If this occurs, the query will fail with the following messages:

```
%RDB−F−IMP_EXC, facility−specific limit exceeded
−RDMS−F−XPR_STACK_OFLO, expression forces too many levels of recursion
```

The size of the executive mode stack can be adjusted using the logical RDMS$BIND_EXEC_STACK_SIZE. The default size is 20 pagelets and the query in the command file referred to above required the size to be set to 330 in order to complete.

This problem has been corrected in Oracle Rdb Release 7.1.5.1.

3.2.5 Unexpected Bugcheck from ALTER VIEW

Bug 5964501

In prior releases of Oracle Rdb, an ALTER VIEW command which specified an AS SELECT clause could bugcheck. This would occur when the column expression included a Boolean expression such as appears in a CASE, COALESCE, NULLIF, NVL, NVL2, ABS, DECODE, or subselect statement.

The following example shows the error.

```
SQL> alter view v_t1 as select nullif(col1,-999999) as nn from t1;
%RDMS-I-BUGCHKDMP, generating bugcheck dump file
RDBVMS_USER2:[SMITHI]RDSBUGCHK.DMP;
%RDMS-I-BUGCHKDMP, generating bugcheck dump file
RDBVMS_USER2:[SMITHI]SQLBUGCHK.DMP;
%SYSTEM-F-ACCVIO, access violation, reason mask=00, virtual address=
000000000000000C, PC=000000000028878C, PS=0000001B
SQL>
```

This problem has been corrected in Oracle Rdb Release 7.1.5.1. In addition, Oracle Rdb now correctly stores the table dependency information in RDB$VIEW_RELATIONS system table for the view.

3.2.6 Unexpected Bugcheck from ALTER INDEX ... BUILD PARTITION

Bug 5951999

In prior releases of Oracle Rdb, it was possible that an ALTER INDEX ... BUILD PARTITION or ALTER INDEX ... REBUILD PARTITION statement on a SORTED RANKED index would bugcheck. This problem does not affect HASHED or SORTED indices.

The bugcheck is initiated by a sanity check within the SORTED RANKED index facility when it detects that database keys for the partition are not sorted within duplicate index keys.

The workaround for this problem is to use the REBUILD ALL PARTITIONS clause or to drop and re-create the index.

This problem has been corrected in Oracle Rdb Release 7.1.5.1. Rdb now correctly sorts the duplicate index key values so that they are ascending by DBKEY.

3.2.7 Unexpected Constraint Activation After ALTER TABLE ... DISABLE CONSTRAINT

In prior releases of Oracle Rdb, the ALTER TABLE command did not enable or disable constraints already active in the current session. Constraints are enabled and disabled using ALTER TABLE ... ENABLE CONSTRAINT constraint-name, ALTER TABLE ... DISABLE CONSTRAINT constraint-name, ALTER TABLE ... ENABLE ALL CONSTRAINTS and ALTER TABLE ... DISABLE ALL CONSTRAINTS.

The following example shows that the constraint is still checked during the session even though it was disabled using ALTER TABLE.

```
SQL> create table my_con3
cont>     (c1 char
cont>     ,c2 char
cont>     ,constraint MYC1 check (c1 in ('a','b')) deferrable);
SQL> commit;
SQL>
SQL> insert into my_con3 (c1,c2) values ('d','e');
1 row inserted
SQL> commit;
```

```
%RDB-E-INTEG_FAIL, violation of constraint MYC1 caused operation to fail
-RDB-F-ON_DB, on database RDB$DEFAULT_CONNECTION
SQL> rollback;
SQL>
SQL> insert into my_con3 (c1,c2) values ('a','b');
1 row inserted
SQL> commit;
SQL> rollback;
SQL>
SQL> alter table my_con3 disable constraint myc1;
SQL> show table (constraint) my_con3;
Information for table MY_CON3

Table constraints for MY_CON3:
MYC1
 Check constraint
 Table constraint for MY_CON3
 Evaluated on COMMIT
 Source:
        CHECK (c1 in ('a','b'))
Constraint is disabled

Constraints referencing table MY_CON3:
 No Constraints found

SQL> commit;
SQL>
SQL> insert into my_con3 (c1,c2) values ('d','e');
1 row inserted
SQL> commit;
%RDB-E-INTEG_FAIL, violation of constraint MYC1 caused operation to fail
-RDB-F-ON_DB, on database RDB$DEFAULT_CONNECTION
```

This problem has been corrected in Oracle Rdb Release 7.1.5.1. Oracle Rdb now correctly applies the enable/disable to the active constraints in the session.

3.2.8 Bugcheck with "SQL$BLRXPR − 15" on Cross−DB Insert

Bug 4307036

In certain cases, an INSERT statement with a cross−database select would fail and produce a SQLBUGCHK.DMP that contains the following line:

```
%SQL-F-BUGCHK, There has been a fatal error. Please contact your Oracle
support representative. SQL$BLRXPR − 15
```

Such INSERT statements would involve a DISTINCT keyword or a CASE expression in the SELECT. For example, consider the following database and variable declarations:

```
$ SQL$
SQL> create database filename temp1;
SQL> create table ctab (
cont> destination char(4),
cont> state       char(2)
cont> );
SQL> insert into ctab values ('ABCD','ME');
```

```
1 row inserted
SQL> insert into ctab values ('EFGH','ME');
1 row inserted
SQL> insert into ctab values ('IJKL','MI');
1 row inserted
SQL> insert into ctab values ('ABCD','ME');
1 row inserted
SQL> commit;
SQL> disconnect all;
SQL> create database filename temp2;
SQL> create table tb1 (
cont> ccode   char(4),
cont> snum    integer
cont> );
SQL> commit;
SQL> disconnect all;
SQL> att 'alias db2 file temp2';
SQL> att 'alias db1 file temp1';
SQL> declare : run_id integer = 5;
SQL> declare :xx char(2) = 'xx';
SQL> declare :suffix char(2) = 'ME';
SQL> declare :hv2 integer = 1;
```

In the above context, the following cross−database INSERT statements would fail as shown:

```
SQL> insert into db2.tb1
cont>  (select distinct destination, :run_id from db1.ctab where state='ME');
%RDMS−I−BUGCHKDMP, generating bugcheck dump file MY$DISK:[MY_HOME]SQLBUGCHK.DMP;
%SQL−F−BUGCHK, There has been a fatal error. Please contact your Oracle support
representative. SQL$BLRXPR − 15
SQL> insert into db2.tb1
cont> (select destination,
cont>  case when :suffix indicator :hv2 is null
cont>       then 1 else 2 end case
cont>  from db1.ctab);
%RDMS−I−BUGCHKDMP, generating bugcheck dump file MY$DISK:[MY_HOME]SQLBUGCHK.DMP;
%SQL−F−BUGCHK, There has been a fatal error. Please contact your Oracle support
representative. SQL$BLRXPR − 15
```

This problem has been corrected in Oracle Rdb Release 7.1.5.1.

3.2.9 Comments Not Always Recognized by SQL$PRE/COBOL

Bug 4751103

The SQL Precompiler was not properly recognizing COBOL comment statements. Most of the time this makes no difference but in certain cases, such as in the middle of a record definition, it was possible for names to be missed resulting in a error such as a %SQL−F−HVNOTDECL.

For example, consider the following COBOL program with embedded SQL.

```
000000 IDENTIFICATION DIVISION.
000000 PROGRAM−ID.    DUMMYIO.
000000 AUTHOR.        RDB ENGINEERING.
000000
000000 ENVIRONMENT DIVISION.
```

```
000000
000000 CONFIGURATION SECTION.
000000 SOURCE-COMPUTER. ALPHA-RDB.
000000 OBJECT-COMPUTER. ALPHA-RDB.
000000
000000 DATA DIVISION.
000000
000000 WORKING-STORAGE SECTION.
000000
000000     EXEC SQL INCLUDE SQLCA END-EXEC
000000
000000 01  HOST-DATA.
000000     02  HOST-CODE                PIC  X(03).
000000* COMPUTATIONAL ITEMS
000000     02  OLD-TOTAL-COST             PIC S9(09)V99    COMP.
000000     02  OLD-TOTAL-TAX              PIC S9(09)V99    COMP.
000000* END OF COMPUTATIONAL ITEMS
000000     02  HOST-REASON                PIC  X(60).
000000     02  USER-CODE                  PIC  X(05).
000000
000000 01  SELECTION-RANGES.
000000     02  NO-RANGE                          PIC  X(1).
000000
000000 PROCEDURE DIVISION.
000000
000000 THE-PROGRAM.
000000
000000     PERFORM SELECT-HOST-RECORD.
000000     MOVE SPACE TO NO-RANGE.
000000
000000 RETURN-TO-CALLER.
000000     EXIT PROGRAM.
000000
000000 SELECT-HOST-RECORD.
000000     EXEC SQL SELECT COUNT(*)
000000     INTO :OLD-TOTAL-COST
000000     FROM RDB$DATABASE
000000     END-EXEC.
```

In the above program, there are two comments inside the definition of the HOST−DATA record. The SQL Precompiler would not successfully parse these comments and, as a result, would give the following error message:

```
000000     INTO :OLD-TOTAL-COST
                1
%SQL-F-HVNOTDECL, (1) Host variable OLD-TOTAL-COST was not declared
```

In the bug report, the embedded comments were a pair of alignment compiler directives which are special COBOL comment lines that appear similar to the following:

```
000000 01  HOST-DATA.
000000     02  HOST-CODE                PIC  X(03).
000000*DC SET ALIGNMENT
000000     02  OLD-TOTAL-COST             PIC S9(09)V99    COMP.
000000     02  OLD-TOTAL-TAX              PIC S9(09)V99    COMP.
000000*DC END-SET ALIGNMENT
000000     02  HOST-REASON                PIC  X(60).
000000     02  USER-CODE                  PIC  X(05).
```

The problem can be worked around by removing the comments from the middle of the record definition. If the comments are alignment compiler directives, such as the ALIGNMENT directives shown above or the PADALIGN directive, these can be repositioned to bracket the entire record and the same COBOL code will be generated. For example, the record definition shown above can be recoded as:

```
000000*DC SET ALIGNMENT
000000 01  HOST-DATA.
000000     02  HOST-CODE                 PIC  X(03).
000000     02  OLD-TOTAL-COST             PIC S9(09)V99    COMP.
000000     02  OLD-TOTAL-TAX              PIC S9(09)V99    COMP.
000000     02  HOST-REASON                PIC  X(60).
000000     02  USER-CODE                  PIC  X(05).
000000*DC END-SET ALIGNMENT
```

This problem has been corrected in Oracle Rdb Release 7.1.5.1.

3.2.10 Unexpected Bugcheck When Inserting Into a View

Bug 4001764

In prior releases of Oracle Rdb V7.1, using a view that referenced an AUTOMATIC AS column or a column with a DEFAULT value could cause a bugcheck. This occurred when the referenced column also referenced other columns from the base table. The following example shows the problem.

```
SQL> create table sample_auto
cont>     (cola integer(3)
cont>     ,colb automatic insert as round(cola));
SQL> insert into sample_auto values (18.245);
1 row inserted
SQL>
SQL> create view sample_view as select * from sample_auto;
SQL>
SQL> insert into sample_view values (19.456);
%RDMS-I-BUGCHKDMP, generating bugcheck dump file USER2:[TESTING]RDSBUGCHK.DMP;
%RDMS-I-BUGCHKDMP, generating bugcheck dump file USER2:[TESTING]SQLBUGCHK.DMP;
%SYSTEM-F-ACCVIO, access violation, reason mask=00,
virtual address=0000000000000010, PC=0000000000288F7C, PS=0000001B
SQL>
```

This problem has been corrected in Oracle Rdb Release 7.1.5.1. Oracle Rdb now correctly resolves the reference to the AUTOMATIC AS column and the DEFAULT value clause from within a view.

3.2.11 SQLSTATE 22001 Not Returned with Dynamic SQL

Bug 5208854

When using the dialects SQL92, SQL99, ORACLE LEVEL1, or ORACLE LEVEL2, SQL is required to report "22001" in the SQLSTATE field along with a negative SQLCODE when a value input into a SQL statement is truncated.

If a Precompiled SQL program used a variable to provide an input value for a SQL statement and this variable was longer than the target database field and contained non−blank data beyond the size of the field, the required SQLSTATE and SQLCODE were not being reported.

For example, consider a database table named CH1 with the following columns defined:

```
Column Name                     Data Type
-----------                     ---------
CH1A                            CHAR(10)
CH1B                            CHAR(1)
CH1C                            CHAR(10)
```

The following fragment of a Precompiled SQL/C program contains the following insert code:

```
...
    long SQLCODE;
    char SQLSTATE[6] = {'\0','\0','\0','\0','\0','\0'};
    char dstmt[100];
    char x4[16];
...
    strcpy(dstmt, "SET DIALECT 'SQL92'");
    EXEC SQL EXECUTE IMMEDIATE :dstmt;

    strcpy (dstmt, "INSERT INTO CH1 VALUES ('FOO', 'F', ?)          ");
    EXEC SQL PREPARE BLAT4 FROM :dstmt;

    strcpy (x4, "LITTLETOOLONG");
    EXEC SQL EXECUTE BLAT4 USING :x4;

    printf("SQLCODE should be < 0, SQLSTATE should be 22001\n")
    printf("SQLCODE is %ld\n", SQLCODE);
    SQLSTATE[5] = '\0';
    printf ("SQLSTATE is %s\n", SQLSTATE);
...
```

The output of this program would have appeared as:

```
SQLCODE should be < 0, SQLSTATE should be 22001
SQLCODE is 0
SQLSTATE is 00000
```

The output of the program should be:

```
SQLCODE should be < 0, SQLSTATE should be 22001
SQLCODE is -306
SQLSTATE is 22001
```

This problem has been corrected in Oracle Rdb Release 7.1.5.1.

3.3 RMU Errors Fixed

3.3.1 RMU/VERIFY/INCREMENTAL Incorrect Diagnostics for Bitmap Indexes

Bug 4149656

Even though RMU/VERIFY knew it was walking a btree bitmap index, a bad flag passed to a routine verifying the index could cause "hashed" to be incorrectly inserted in the message for the index type. Also, invalid bitmap btree index diagnostics such as RMU−W−BADHDADBK, RMU−I−BTRDUPCAR, and RMU−I−BTRERPATH could be output by RMU/VERIFY/INCREMENTAL even though the index did not have any problems. This was caused by a loss of context while RMU/VERIFY was validating the btree index duplicate bitmap chain.

This problem only happened when RMU/VERIFY/INCREMENTAL was verifying btree bitmap index duplicate chains. It did not happen if /NODATA was specified. It happened on longer bitmap chains that required fetching multiple duplicate bitmap records.

The following example shows invalid index diagnostics output by RMU/VERIFY/INCREMENTAL.

```
$ RMU/VERIFY/INCREMENTAL/ALL TEST_DATABASE
%RMU−W−BADHDADBK, Bad logical area DBID in hashed index data record
dbkey 1716:674953227:−32678 Found 06B4 (hex).
%RMU−W−BADHDADBK, Bad logical area DBID in hashed index data record dbkey
1716:674953227:−32677 Found 06B4 (hex).
%RMU−W−BADHDADBK, Bad logical area DBID in hashed index data record dbkey
1716:674953227:−32676 Found 06B4 (hex).
```

A workaround for this problem is to either specify /NODATA or to not specify /INCREMENTAL for the verify.

```
$ RMU/VERIFY/INCREMENTAL/INDEX/NODATA TEST_DATABASE
$ RMU/VERIFY/ALL TEST_DATABASE
```

This problem has been corrected in Oracle Rdb Release 7.1.5.1.

3.3.2 RMU LOAD Delimited Text Problem Parsing NULL Values

Bug 5768090

There was a problem parsing values flagged as NULL in RMU UNLOAD and LOAD using DELIMITED TEXT delimiters that occurred when the character string used to designate a field flagged as NULL was the same as the combined character strings used to designate the PREFIX and SUFFIX delimiters which mark the start and end of column values. This problem was due to RMU/LOAD interpreting the NULL value as a PREFIX delimiter followed by a SUFFIX delimiter. This caused the column not to be flagged as NULL when it was loaded. This could cause errors on the RMU/LOAD. One example is an attempt to load an invalid date value if the field being loaded was a date field. It could also cause unintended values to be loaded to a field

that was supposed to be flagged as NULL.

This problem has been fixed but we strongly recommend that a unique string or single character value for NULL be used that cannot be confused by the RMU UNLOAD and LOAD parsing with the prefix, the suffix or the prefix plus suffix single character or string values.

In ambiguous cases, where identical delimiters can be confused, the parser has to give precedence to one choice over another. It tries to pick the most common and reasonable case, but that may not always coincide with the original intent of the user. Using a unique NULL value or taking the default NULL value would avoid these cases.

The following shows one example of the case we have fixed where a problem occurred during the RMU/LOAD because the NULL string equaled the combined values of the prefix and the suffix character strings, which in this case each consisted of a single character. This case might not be obvious to a user because the prefix and suffix characters were not specified but the documented default prefix and suffix single character delimiter values of " were used. Since the NULL string was specified as "", RMU/LOAD interpreted the NULL value "" as a default prefix value " followed by a default suffix value ". This caused the NULL bit not to get set for the middle columns. Note that as documented

```
NULL="""""""
```

designates a NULL designator value of "" since the two outer double quote characters are stripped and each " character must be entered as "".

```
$ SQL
create database filename test;
create table t1 (col1 char(5), col2 date vms, col3 char(5));
create table t2 (col1 char(5), col2 integer, col3 char(5));
create table t3 (col1 char(5), col2 char(5), col3 char(5));
commit;
insert into t1 (col1,col3) values ('first','last');
insert into t2 (col1,col3) values ('first','last');
insert into t3 (col1,col3) values ('first','last');
commit;
SQL> sel * from t1;
 COL1    COL2                      COL3
 first   NULL                      last
1 row selected
SQL> sel * from t2;
 COL1           COL2   COL3
 first          NULL   last
1 row selected
SQL> sel * from t3;
 COL1    COL2    COL3
 first   NULL    last
1 row selected
 Exit
$ rmu/unload/record=(file=t1,format=delimited,null="""""""") test t1 t1
$ rmu/unload/record=(file=t2,format=delimited,null="""""""") test t2 t2
$ rmu/unload/record=(file=t3,format=delimited,null="""""""") test t3 t3
$ type t1.unl
"first","","last "
$ type t2.unl
"first","","last "
$ type t3.unl
"first","","last "
```

Because RMU confused the NULL value for the PREFIX plus SUFFIX values, the RMU/LOAD fails for the date field, and inserts a zero in the case of an integer (or real), and an empty string in the case of a character field.

```
$ rmu/load/record=(file=t1,format=delimited,null="""""""") test t1 t1
%RMU-I-LOADERR, Error loading row 1.
%RDB-E-CONVERT_ERROR, invalid or unsupported data conversion
-COSI-F-IVTIME, invalid date or time
%RMU-I-DATRECREAD,  1 data records read from input file.
%RMU-I-DATRECSTO,   0 data records stored.
%RMU-F-FTL_LOAD, Fatal error for LOAD operation at 18-JAN-2007 04:23:54.48
$ rmu/load/record=(file=t2,format=delimited,null="""""""") test t2 t2
%RMU-I-DATRECREAD,  1 data records read from input file.
%RMU-I-DATRECSTO,   1 data records stored.
$ rmu/load/record=(file=t3,format=delimited,null="""""""") test t3 t3
%RMU-I-DATRECREAD,  1 data records read from input file.
%RMU-I-DATRECSTO,   1 data records stored.
$ SQL
SQL> att 'f test';
SQL> sel * from t1;
 COL1    COL2                      COL3
 first   NULL                      last
1 row selected
SQL> sel * from t2;
 COL1           COL2   COL3
 first          NULL   last
 first             0   last
2 rows selected
SQL> sel * from t3;
 COL1    COL2    COL3
 first   NULL    last
 first           last
2 rows selected
```

A workaround for this problem is to designate a unique value for NULL that cannot be confused with the default or specified values of other DELIMITED TEXT delimiters.

This problem has been corrected in Oracle Rdb Release 7.1.5.1.

3.3.3 %COSI−F−NEGTIM Could Abort RMU/VERIFY Or RMU/BACKUP

Bug 6005440

There was a problem where the following message

```
%COSI-F-NEGTIM, a negative time was computed
```

could abort the RMU/BACKUP or RMU/VERIFY of an Oracle Rdb database. This happened during the Oracle Rdb database root verification executed by the RMU/VERIFY and RMU/BACKUP commands if the Oracle Rdb database root creation time was outside the allowable range. The %COSI−F−NEGTIM error was allowed to abort the RMU/BACKUP or RMU/VERIFY operation when the command execution should have been allowed to continue.

One possible cause of this problem besides database root corruption would be if the VMS system time was set to a time earlier than the database creation time.

This problem has been corrected so that if a %COSI−F−NEGTIM error does occur when the database root is verified, the RMU/VERIFY or RMU/BACKUP will be allowed to continue.

The following example of this problem happens during the verification of the database root at the beginning of the backup of an Oracle Rdb database. After the %RMU−W−ROOTADINV warning is output because of an invalid creation date value in the database root, the %COSI−F−NEGTIM error aborts the backup operation.

```
$rmu/backup mf_personnel mfp.rbf
%RMU−W−ROOTADINV, root "DEVICE:[DIRECTORY]MF_PERSONNEL.RDB;1",
                  contains incorrect time stamp
                  expected between  1−DEC−1981 00:00:00.00 and
                  1−JAN−2007 00:00:01.24,
%COSI−F−NEGTIM, a negative time was computed
%RMU−F−FATALOSI, Fatal error from the Operating System Interface.
%RMU−F−FTL_BCK, Fatal error for BACKUP operation at  1−JAN−2007 00:00:02.59
```

If this problem is caused by the VMS system time being set to a time earlier than the database creation time, a workaround for this problem would be to set the system time to a time after the database creation. Otherwise, RMU/RESTORE/ONLY_ROOT could be used to restore a valid database root from a previous database backup.

This problem has been corrected in Oracle Rdb Release 7.1.5.1.

3.3.4 ACCVIO from Non−valid Response to RMU Prompt

Bug 6001187

Responding to a "QUIT or CONTINUE" RMU prompt with an invalid answer causes an ACCVIO. This can be seen in the following example.

```
$ RMU/DUMP/BACK MFP.RBF
%RMU−I−DMPTXT_163, No dump option selected. Performing read check.
%RMU−E−READERR, error reading USER:[USERDIR]MFP.RBF;
−RMU−E−BLOCKCRC, software block CRC error
%RMU−E−INVBLKSIZ, invalid block size in backup file
%RMU−E−INVRECSIZ, invalid record size in backup file
%RMU−E−READERRS, excessive error rate reading USER:[USERDIR]MFP.RBF;
−RMU−E−BLOCKCRC, software block CRC error
%SYSTEM−F−ACCVIO, access violation, reason mask=00, virtual address=...
```

The reason for this is that there were incorrect variables in two places in the code.

This problem has been corrected in Oracle Rdb Release 7.1.5.1.

3.3.5 RMU/RESTORE Exits with Traceback Log when the Wrong Version is Used

Bug 6001235

Using RMU/RESTORE to restore an incompatible version of a database exits with a register dump instead of simply exiting. This can be seen in the following example.

```
$ RMU/RESTORE/NOCDD/DIR=SYS$DISK:[]/LOG REPRODUCER.RBF
%RMU-F-DB_CVT_FAIL, Cannot convert from version V7.2 to V7.1
%RMU-F-DB_CVT_FAIL, Cannot convert from version V7.2 to V7.1

  Improperly handled condition, image exit forced by last chance handler.
    Signal arguments:   Number = 0000000000000008
                        Name   = 0000000002C88B94
                                 0000000000000004
                                 0000000000000007
                                 0000000000000002
                                 0000000000000007
                                 0000000000000001
                                 00000000003AEB34
                                 100000000000001B
  ...
```

This problem has been corrected in Oracle Rdb Release 7.1.5.1.

3.3.6 RMU/VERIFY %RMU−I−BADNXTNOD Diagnostic Message Changed To %RMU−E−BADNXTNOD

Bug 4197298

The %RMU−I−BADNXTNOD diagnostic message that could be put out when verifying Oracle Rdb database sorted indexes indicated that the index structure was corrupt and that the index needed to be rebuilt. Therefore the BADNXTNOD diagnostic message severity level has been changed from Informational to Error so that this message will not be ignored.

The following example shows the %RMU−I−BADNXTNOD diagnostic message that was put out when RMU/VERIFY detected a corruption in the structure of a sorted index in an Rdb database. This indicated that the index needed to be rebuilt.

```
$ RMU/VERIFY/INDEX=bad_index mf_personnel
%RMU-I-BADNXTNOD, Bad next b-tree node at level 2.
                  Expected b-tree node at logical dbkey 39:3:2.
                  Found next b-tree node at logical dbkey 65:3:2.
```

The following example shows the %RMU−E−BADNXTNOD diagnostic message which will now be output with an "Error" instead of an "Informational" severity to make sure that the index corruption is brought to the user's attention.

```
$ RMU/VERIFY/INDEX=bad_index mf_personnel
%RMU-E-BADNXTNOD, Bad next b-tree node at level 2.
                  Expected b-tree node at logical dbkey 39:3:2.
                  Found next b-tree node at logical dbkey 65:3:2.
```

This problem has been corrected in Oracle Rdb Release 7.1.5.1.

3.4 RMU Show Statistics Errors Fixed

3.4.1 RMU/SHOW STATISTICS AIJ ARB:I/O Ratio, Blocks−per−I/O Ratio Problems

There was a problem with detecting the warning thresholds set for the "Examine ARB:I/O ratio" and "Examine blocks−per−I/O ratio" options on the "AIJ Analysis" screen display produced by the RMU/SHOW STATISTICS command for Oracle Rdb databases with after image journaling enabled. This problem caused the "ARB:I/O ratio" warning

```
#.# ARBs per I/O below #.# threshold
```

to not always be output when the currently set threshold for this ratio was passed and caused the "blocks−per−I/O ratio" warning

```
#.# blocks written per I/O below #.# threshold
```

to not always be output when the currently set threshold for this ratio was passed. This problem was caused by treating these thresholds as percent values instead of count values. This problem has been fixed and the misleading percent signs have been removed from these warning messages.

This problem has been corrected in Oracle Rdb Release 7.1.5.1.

3.4.2 State Value Truncated on Hot Standby Statistics Display

Bug 6044632

Previously, it was possible when using a Hot Standby TCP/IP port number greater than 9999 that the port number would be truncated on the RMU /SHOW STATISTICS Hot Standby Statistics display.

For example, when using a TCP/IP port number of 12345, the state display could be shown as "State: TCP/IP:1234"

This problem has been corrected in Oracle Rdb Release 7.1.5.1. The state display field now allows a 5−digit TCP/IP port number to be displayed.

3.5 Hot Standby Errors Fixed

3.5.1 Hot Standby Node Failure Recovery When Using RMU/OPEN/ROW_CACHE=DISABLE

Bug 5957364

In configurations using the Row Cache and Hot Standby features, row caching must be explicitly disabled on the standby database using the RMU/SET ROW_CACHE/DISABLE command prior to starting Hot Standby for the first time on the database. However, it is also possible (though not recommended) to use the RMU/OPEN/ROW_CACHE=DISABLE command on the standby database in order to suppress row caching.

When using the RMU/OPEN/ROW_CACHE=DISABLE command, if a system failure occurred, it was possible that the database recovery upon reopening the database would attempt to start with a very old last checkpoint location. This location was based on the row cache checkpoint from when the master database had been originally backed up to create the standby database. In some cases, the required AIJ files would no longer be online and the recovery would fail.

This problem has been corrected in Oracle Rdb Release 7.1.5.1. The DBR process now ignores the row cache oldest checkpoint location when not recovering from a node failure when the RCS process had been active.

Chapter 4
Software Errors Fixed in Oracle Rdb Release 7.1.5

This chapter describes software errors that are fixed by Oracle Rdb Release 7.1.5.

4.1 Software Errors Fixed That Apply to All Interfaces

4.1.1 Bugcheck Loop Created Many Bugcheck Dumps

Bug 5411895

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

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

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

This change has been made in Oracle Rdb Release 7.1.5.

4.1.2 Left Outer Join Query Slows With Full Index Scan

Bug 5567495

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

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

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

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

create view v_t1_loj_t2
    (col_one, col_two, total) as
    (select c2.col_one, c2.col_two, SUM(c3.line_amt)
    from t1 as c2
       LEFT OUTER JOIN
         t2 as c3
```

```
            ON ((c3.col_one = c2.col_one) AND
                (c3.col_two = c2.col_two))
    GROUP BY c2.col_one, c2.col_two);

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

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

The strategy from Rdb Release 7.1.4.1.1 is exactly the same with the exception of the better performing index retrieval in the outer loop of the left outer join operation.

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

There is no known workaround for this problem.

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

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

This problem has been corrected in Oracle Rdb Release 7.1.5.

4.1.3 Wrong Result From Query With Zig−zag Match and Reverse Scan

Bugs 5842319, 5850013, and 4771936

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

```
SQL> set flags 'strategy,detail';
SQL> select d.trx_id
cont>  from t1 d
cont>  where exists (select * from t2 where trx_id = d.trx_id);
Tables:
  0 = T1
  1 = T2
Conjunct: <agg0> <> 0
Match
  Outer loop      (zig−zag)
    Index only retrieval of relation 0:T1
      Index name  T1_NDX [0:0]
  Inner loop      (zig−zag)
    Aggregate−F1: 0:COUNT−ANY (<subselect>)
    Index only retrieval of relation 1:T2
      Index name  T2_NDX [0:0]      Reverse Scan
0 rows selected
```

The tables contain the following data.

```
SQL> select * from t1;
 TRX_ID    SEQUENCE_NO
 0000044             1
 0000046             2
 0000047             1

SQL> select * from t2;
 TRX_ID    LOCATION_ID    COMPANY_NO
 0000046             5             2
 0000045             5             2
```

A workaround is to use SET FLAGS or define RDMS$SET_FLAGS to the value NOREVERSE_SCAN to disable the reverse scan feature. See the following example.

```
SQL> set flags 'noreverse_scan'
SQL> !... execute the same above query here ...
Tables:
  0 = T1
  1 = T2
```

```
Cross block of 2 entries
  Cross block entry 1
    Index only retrieval of relation 0:T1
      Index name  T1_NDX [0:0]
  Cross block entry 2
    Conjunct: <agg0> <> 0
    Aggregate-F1: 0:COUNT-ANY (<subselect>)
    Index only retrieval of relation 1:T2
      Index name  T2_NDX [1:1]
        Keys: 1.TRX_ID = 0.TRX_ID
 TRX_ID    SEQUENCE_NO
 0000046             2
1 row selected
```

This is problem is related to an incomplete fix for Bug 4771936. The key parts of this query which contributed to the error are:

1. The main select query joins two tables (in this example using an EXISTS clause) and a zig−zag match strategy is chosen for the solution.
2. The join key for the inner leg of the join is based on an index with DESC (descending) index segment but an ascending segment in the outer index.
3. A reverse scan is applied on the index retrieval at the inner leg.

This problem may occur under similar conditions for NOT EXISTS and normal join queries.

This problem affects the following Oracle Rdb Releases: V7.1.4.4, V7.1.4.5, V7.2.0.1, V7.2.0.2 and V7.2.1. This problem has been corrected in Oracle Rdb Release 7.1.5.

4.1.4 RDB−E−EXCESS_TRANS Error After SET TRANSACTION Failure

Bug 5755008

Under some circumstances, a failure in a SET TRANSACTION statement with multiple databases, including at least one remote database, would leave that remote database in an indeterminate state. In this state, all SQL statements received an %RDB−E−EXCESS_TRANS error which named the remote database as having a transaction already in progress. The following ingredients were necessary to encounter the problem:

- Multiple databases must be attached, including at least one remote database.
- A transaction must be started for which Rdb does not use 2−phase commit. (For example, if all databases are read−only or if the logical SQL$DISABLE_CONTEXT is defined as TRUE, Rdb does not start a distributed transaction.)
- Rdb succeeds in starting a transaction on the remote database and then fails in starting the transaction on some other attached database.

In the above circumstance, Rdb will rollback the transaction for each of the databases for which it successfully started a transaction before failing on one of the databases. The most common reason for such a failure is a lock conflict with another process.

The following example illustrates the failure:

```
SQL> ATTACH 'ALIAS A1 FILENAME XENA::PERSONNEL';
```

```
SQL> ATTACH 'ALIAS A2 FILENAME PERSONNEL';
SQL> SET TRANSACTION ON A1 USING
cont>    (READ ONLY RESERVING A1.employees FOR SHARED READ)
cont>            AND ON A2 USING
cont>    (READ ONLY RESERVING A2.employees FOR SHARED READ,
cont>                         A2.bogus FOR SHARED READ);
%RDB-E-OBSOLETE_METADA, request references metadata objects that no longer exist
-RDMS-F-RELNOEXI, relation BOGUS
SQL>
SQL> show table a2.employees
%RDB-E-EXCESS_TRANS, exceeded limit of 1 transaction per database attachment
-RDB-F-ON_DB, on database DISK:[DIR]MF_PERSONNEL.RDB;1
```

In the above example, alias A1 is a remote database and alias A2 is a local database. Because the SET TRANSACTION statement specifies READ ONLY for both databases, Rdb does not use a distributed transaction (this is an optimization). The SET TRANSACTION statement fails because it tries to reserve a table named "bogus" in alias A2 which does not exist. This failure is normal and expected. But after the failure of the SET TRANSACTION, Rdb didn't properly roll back the transaction on alias A1. This caused the failure of the subsequent SHOW statement and all subsequent statements with an RDB−E−EXCESS_TRANS error.

The problem can be worked around by naming the remote alias last in the SET TRANSACTION statement.

This problem has been corrected in Oracle Rdb Release 7.1.5.

4.2 SQL Errors Fixed

4.2.1 Unexpected Constraint Failure from INSERT ... SELECT Statement

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

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

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

This problem has been corrected in Oracle Rdb Release 7.1.5. If the SQL language dialect SQL92, SQL99, ORACLE LEVEL1 or ORACLE LEVEL2 is set, the NOT DEFERRABLE constraint evaluation is now performed after all rows have been inserted for the INSERT ... SELECT statement.

4.2.2 Numeric Out of Range SQLSTATE 22003 Not Returned

Bug 5208860

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

For example, consider a table defined as follows:

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

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

```
SQL> INSERT INTO NUM1 VALUES (−10, 0);
%RDB−E−VALOUTRANGE, value outside the specified precision (3) for column
"NUM1C1"
SQL> show sqlca
SQLCA:
        SQLCAID:        SQLCA           SQLCABC:        128
        SQLCODE:        −1
        SQLERRD:        [0]: 2
                        [1]: 0
                        [2]: 0
                        [3]: 0
                        [4]: 0
                        [5]: 0
        SQLWARN0:       0       SQLWARN1:       0       SQLWARN2:       0
        SQLWARN3:       0       SQLWARN4:       0       SQLWARN5:       0
        SQLWARN6:       0       SQLWARN7:       0
        SQLSTATE:       RR000
```

This INSERT now produces the following results:

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

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

This problem has been corrected in Oracle Rdb Release 7.1.5.

4.2.3 TIMESTAMP Result of DATE Added to TIME Expression was Truncated

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

The following example shows that the fractional seconds were truncated.

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

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

This problem has been corrected in Oracle Rdb Release 7.1.5. SQL now correctly assigns the fractional seconds precision of the TIME value expression to the TIMESTAMP result.

4.2.4 CREATE OUTLINE Not Fully Supported for MULTISCHEMA Databases

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

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

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

```
cont>       )
cont>     )
cont>   )
cont> compliance optional     ;
%RDB-E-NO_META_UPDATE, metadata update failed
-RDMS-F-INDNOTEXI, index T_INDEX does not exist in this database
```

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

These problems have been corrected in Oracle Rdb Release 7.1.5.

4.2.5 Unexpected INV_TBL_DCL Error From CREATE MODULE in Compiled Source

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

The following example shows these errors.

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

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

This problem has been corrected in Oracle Rdb Release 7.1.5.

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

Bug 5620530

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

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

```
$ DEFINE/USER RDMS$SET_FLAGS -
    "TEST_SYSTEM,STOMAP_STATS,INDEX_STAT,INTERNAL,ITEM_LIST"
$ SQL$
alter database
    filename SAMPLE_DATABASE
```

```
    drop storage area AREA_MAIN_07 cascade;
~As: Drop Storage Area "AREA_MAIN_07" Cascade
~As: ...area referenced by index: "INDEX1"
~As: ...area referenced by map: "MAP_TABLE_001"
~As: ...purge index "INDEX2"
~As: ...update the AIP for larea=65 (table)
~As: ...update the AIP for larea=77 (index)
~H Extension (VERIFY CONSTRAINTS) Item List:
0000 (00000) RDB$K_EXT_VFYC_EXCLUDE_UNIQUE
0003 (00003) RDB$K_EXT_VFYC_TABLE_NAME "TABLE_001"
000F (00015) RDB$K_INFO_END
$
$ RMU/VERIFY/INDEX SAMPLE_DATABASE
%RMU-W-CANTFINDLAREA, cannot locate logical area 65 in area inventory page list
%RMU-E-BDLAREADY, error readying logical area with dbid 65
%RMU-I-READYDATA, ready needed for data record at 65:5:0
%RMU-I-BTRNODDBK, Dbkey of B-tree node is 89:3:0
%RMU-W-BTRVFYPRU, B-tree verification pruned at this dbkey
%RMU-I-BTRPARROO, root dbkey of b-tree partition in AREA_INDEX_07 is 89:3:0
$
```

This problem has been corrected in Oracle Rdb Release 7.1.5. The only workaround for this problem is to drop and recreate the affected indices.

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

Bug 5629307

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

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

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

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

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

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

This problem has been corrected in Oracle Rdb Release 7.1.5. SQL now uses the octet length of the maximum size character in the character set for the estimation. While it is now less likely that TRANSLATE will issue unnecessary LENMISMAT warnings, SQL may not know the final translation of the source character string, and for some variable length character sets the warning may be justified even when the assignment succeeds without truncation.

4.2.8 Unexpected Error from UNION Containing NULL Expression

Bug 5645199

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

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

The following example shows the incorrect results.

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

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

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

This problem has been corrected in Oracle Rdb Release 7.1.5.

4.2.9 Inconsistent Data Type Assignment to IS NULL Expression

Bug 5484239

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

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

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

This problem has been corrected in Oracle Rdb Release 7.1.5. In this release, SQL will assign the default type (that is VARCHAR(2000)) as the data type for the expression "? IS NULL".

4.2.10 Table Synonym Not Used by Query Outlines

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

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

This problem has been corrected in Oracle Rdb Release 7.1.5. Oracle Rdb now compares the result target table instead of just comparing the name referenced in the query outline with that of the table in the query.

4.2.11 Unexpected UNSDTPCVT Error During String Concatenation

Bug 5584169

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

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

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

This problem has been corrected in Oracle Rdb Release 7.1.5.

4.2.12 Parameter for LIKE Pattern Sized Too Small

Bug 4179408

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

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

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

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

```
$ run test$tools:tester
Enter statement:
attach 'filename sql$database';
Enter statement:
create table LL (val char(4));
Enter statement:
insert into LL (val) values ('1234');
Enter statement:
insert into LL (val) values ('1235');
Enter statement:
insert into LL (val) values ('0123');
```

```
Enter statement:
select * from LL where val like ?;
[1 fields]
 0/VAL/Varchar(8/12): %123%
 0/VAL: 1234
 0/VAL: 1235
 0/VAL: 0123
Enter statement:
```

This problem has been corrected in Oracle Rdb Release 7.1.5. SQL now assumes that the like pattern is twice the size of the source string, or if the ESCAPE clause is present, it assumes three times the size. This should allow room for most pattern strings. If this sizing is still too small, use CAST(? AS VARCHAR(n)) to size the parameter to a more precise length.

4.2.13 SQL/Services Executor Loops Consuming 99% CPU

Bugs 4401924 and 5353228

After upgrading to SQL/Services 7.1.5.9.1 with Rdb 7.1.4.3.1 and later versions, some applications occasionally experienced a SQL/Services executor which enters a tight loop consuming a very high percentage of the CPU until it is stopped. This problem typically happens immediately after an executor begins servicing a new client and severely degrades the performance of everything else on the VMS node where it occurs. The only way to clear the problem is to STOP/ID the looping executor process.

The problem was caused by memory corruption associated with SQL's management of cached metadata. The memory corruption was associated with multiple SQL connections where some connection had cursors defined. It usually occurred after a DISCONNECT statement but might occur at other times. As such, it could affect any application which uses SQL connections and cursors, not just a SQL/Services executor as described above. Depending on what memory was corrupted, a wide variety of symptoms is also possible including failures in other layers of Rdb as well as in the customer application.

There is no workaround for this problem.

This problem has been corrected in Oracle Rdb Release 7.1.5.

4.2.14 SET AUTOMATIC TRANSLATION 'ON' May Cause Wrong Results From Queries

Bug 5849845

In prior releases of Oracle Rdb V7.1, the SET AUTOMATIC TRANSLATION 'ON' command could cause some queries to return the wrong results. This occurred when a text string literal, parameter or variable was compared with a numeric column. AUTOMATIC TRANSLATION was erroneously processing the non−text values.

The following example shows that the result is incorrect when the SET AUTOMATIC TRANSLATION 'ON' command is used in the session.

```
SQL> set automatic translation 'off';
SQL> select * from TEST_NLS_D where num = '333.333';
%SQL−I−NUMCMPTXT, Numeric column will be compared with string literal as text
```

```
             NUM   TDATE
         333.333    3-MAR-2099 00:00:00.00
1 row selected
SQL>
SQL> set automatic translation 'on';
SQL> select * from TEST_NLS_D where num = '333.333';
%SQL-I-NUMCMPTXT, Numeric column will be compared with string literal as text
0 rows selected
SQL> rollback;
```

This problem has been corrected in Oracle Rdb Release 7.1.5. The default is 'OFF' for most Oracle Rdb users. However, recent versions of OCI Services for Rdb use this command and therefore queries through OCI applications may be impacted.

4.3 RMU Errors Fixed

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

Bug 5464971

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

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

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

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

This problem has been corrected in Oracle Rdb Release 7.1.5.

4.3.2 Incomplete Multischema Database Support in RMU Extract

Bugs 5558122, 5568079, and 5568040

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

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

These problems have been corrected in Oracle Rdb Release 7.1.5.

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

Bugs 5666305 and 5672460

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

This example shows the original view definition.

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

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

```
   .
   .
   .
create view SAMPLE_VIEW
    (A,
     B,
     C) as
    select C3.LAST_NAME, C3.FIRST_NAME, C3.MIDDLE_INITIAL
        * from CANDIDATES C3
        group by C3.LAST_NAME, C3.FIRST_NAME, C3.MIDDLE_INITIAL
    union
    select C5.LAST_NAME, C5.FIRST_NAME, C5.MIDDLE_INITIAL
        * from EMPLOYEES C5
        group by C5.LAST_NAME, C5.FIRST_NAME, C5.MIDDLE_INITIAL;
   .
   .
   .
```

This problem has been corrected in Oracle Rdb Release 7.1.5.

4.4 LogMiner Errors Fixed

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

Bug 5617814

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

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

```
"C",
"",
    .
    .
    .
"16",
"7169",
"00000000-0000-0000-0000-000000000000",
"000000000000038F000000000000038F0000001600000000000001C01"
```

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

This problem has been corrected in Oracle Rdb Release 7.1.5. The correct value for the AERCP_LEN field is now returned. Note that the AERCP_LEN value is 28 and this represents the unformatted binary length of the AERCP structure and not the length of the text formatted field.

4.5 Row Cache Errors Fixed

4.5.1 Bugcheck During Online RMU Backup When Snapshots In Row Cache Enabled

Bug 4260102

In rare situations when using the "snapshots in row cache" feature, a cached database row may be modified to become larger. This modification may require allocation of space on the database page. In this case, it is possible that a row snapshot is not written to the database but rather maintained solely in the row cache. An online RMU operation (such as RMU /BACKUP) may be unable to locate the snapshot row copy and may then fail with a BADPAGNUM, PAGE 4294967295 error in the dump file.

This problem has been corrected in Oracle Rdb Release 7.1.5. The snapshot chain for the row that is having its size adjusted is written to the snapshot storage area before the actual on−disk row modification is made.

4.6 RMU Show Statistics Errors Fixed

4.6.1 Latch Hangs Possible From RMU /SHOW STATISTICS

Bugs 4397634 and 5842040

In prior releases of Oracle Rdb, it was possible in a very small timing window for processes running the RMU /SHOW STATISTICS command to become hung while manipulating "latches" during the database attach sequence. Depending on the exact timing and sequence of events, this process might block other users of the database.

This problem has been corrected in Oracle Rdb Release 7.1.5.

4.7 Hot Standby Errors Fixed

4.7.1 LRS Shutdown Failure RDMS−F−PARTDTXNERR/SYSTEM−F−NOSUCHID

Bug 5754461

A possible problem with the Oracle Rdb Hot Standby feature has been identified. If the OpenVMS $GETGTI system service returns a status value of SS$_NOSUCHID, the LRS process might be unable to shutdown cleanly. This could result in an inconsistent standby database.

This problem has been corrected in Oracle Rdb Release 7.1.5. The LRS process now treats a returned SS$_NOSUCHID status the same as a SS$_NOSUCHTID status and it will be handled normally and will not cause the LRS to fail.

Chapter 5
Enhancements Provided in Oracle Rdb Release 7.1.5.1

5.1 Enhancements Provided in Oracle Rdb Release 7.1.5.1

5.1.1 RMU Tape Support Added for SDLT600, LTO2, LTO3 Drives

Oracle Rdb RMU support has been added for the VMS tape density and compaction values for the Super DLT600, Ultrium460 and Ultrium960 tape drives. This will allow the following new density values to be specified with the /DENSITY qualifier for RMU commands that write to Super DLT600, Ultrium460 and Ultrium960 drives.

```
/DENSITY = (SDLT600,[NO]COMPACTION) - Super DLT600
/DENSITY = (LTO2,[NO]COMPACTION) - Ultrium460
/DENSITY = (LTO3,[NO]COMPACTION) - Ultrium960
```

The following shows examples of specifying density with or without compaction when backing up an Rdb database to one of these tape drives.

```
$ RMU/BACKUP/DENSITY=SDLT600/REWIND/LABEL=(LABEL1,LABEL2) -
 MF_PERSONNEL TAPE1:MFP.BCK, TAPE2:
$ RMU/BACKUP/DENSITY=(SDLT600,COMPACTION)/REWIND/LABEL=(LABEL1,LABEL2) -
 MF_PERSONNEL TAPE1:MFP.BCK, TAPE2:
$ RMU/BACKUP/DENSITY=LTO2/REWIND/LABEL=(LABEL1,LABEL2) -
 MF_PERSONNEL TAPE1:MFP.BCK, TAPE2:
$ RMU/BACKUP/DENSITY=(LTO2,COMPACTION)/REWIND/LABEL=(LABEL1,LABEL2) -
 MF_PERSONNEL TAPE1:MFP.BCK, TAPE2:
$ RMU/BACKUP/DENSITY=LTO3/REWIND/LABEL=(LABEL1,LABEL2) -
 MF_PERSONNEL TAPE1:MFP.BCK, TAPE2:
$ RMU/BACKUP/DENSITY=(LTO3,COMPACTION)/REWIND/LABEL=(LABEL1,LABEL2) -
 MF_PERSONNEL TAPE1:MFP.BCK, TAPE2:
```

5.1.2 New RMU VERIFY Messages %RMU−E−BADCLTSEQALLOC, %RMU−E−BADCLTSEQMAXID, %RMU−E−BADCLTSEQUSED

Three new diagnostic messages have been added to RMU/VERIFY for detecting Oracle Rdb database corruption when verifying Client Sequences. These messages will be output for inconsistencies detected between the client sequence definitions in the database root and the client sequence definitions in the RDB$SEQUENCES system table.

The %RMU−E−BADCLTSEQALLOC message is output if there is an inconsistency between the number of client sequences allocated in the database root and the number of client sequences defined in the system table RDB$SEQUENCES.

The %RMU−E−BADCLTSEQMAXID message is output if there is an inconsistency between the number of client sequences allocated in the database root and the maximum Sequence ID value defined in the system

table RDB$SEQUENCES.

The %RMU−E−BADCLTSEQUSED message is output if there is an inconsistency between the number of client sequences in use in the database root and the number of client sequences defined in the system table RDB$SEQUENCES.

The following example shows all three of these new messages. The %RMU−E−NOSEQENT message is not a new message but an existing message already output by RMU/VERIFY.

```
$ RMU/VERIFY/ALL DISK:[DIRECTORY]MF_PERSONNEL
%RMU-E-BADCLTSEQALLOC, 32 client sequences allocated in the root is less
 than 55 client sequences defined in RDB$SEQUENCES.
%RMU-E-BADCLTSEQMAXID, 32 client sequences allocated in the root is less
 than the maximum client sequence id of 55 in RDB$SEQUENCES.
%RMU-E-NOSEQENT, sequence id 33 has no valid entry in the root file
%RMU-E-NOSEQENT, sequence id 34 has no valid entry in the root file
%RMU-E-NOSEQENT, sequence id 35 has no valid entry in the root file
%RMU-E-NOSEQENT, sequence id 36 has no valid entry in the root file
%RMU-E-NOSEQENT, sequence id 37 has no valid entry in the root file
%RMU-E-NOSEQENT, sequence id 38 has no valid entry in the root file
%RMU-E-NOSEQENT, sequence id 39 has no valid entry in the root file
%RMU-E-NOSEQENT, sequence id 40 has no valid entry in the root file
%RMU-E-NOSEQENT, sequence id 41 has no valid entry in the root file
%RMU-E-NOSEQENT, sequence id 42 has no valid entry in the root file
%RMU-E-NOSEQENT, sequence id 43 has no valid entry in the root file
%RMU-E-NOSEQENT, sequence id 44 has no valid entry in the root file
%RMU-E-NOSEQENT, sequence id 45 has no valid entry in the root file
%RMU-E-NOSEQENT, sequence id 46 has no valid entry in the root file
%RMU-E-NOSEQENT, sequence id 47 has no valid entry in the root file
%RMU-E-NOSEQENT, sequence id 48 has no valid entry in the root file
%RMU-E-NOSEQENT, sequence id 49 has no valid entry in the root file
%RMU-E-NOSEQENT, sequence id 50 has no valid entry in the root file
%RMU-E-NOSEQENT, sequence id 51 has no valid entry in the root file
%RMU-E-NOSEQENT, sequence id 52 has no valid entry in the root file
%RMU-E-NOSEQENT, sequence id 53 has no valid entry in the root file
%RMU-E-NOSEQENT, sequence id 54 has no valid entry in the root file
%RMU-E-NOSEQENT, sequence id 55 has no valid entry in the root file
%RMU-E-BADCLTSEQUSED, 32 client sequences in use in the root does not
 equal 55 client sequences defined in RDB$SEQUENCES.
```

All three of these messages show database corruption that will require a database restore and recovery of the database to the last state that does not show this corruption.

5.1.3 Sample of Rdb External Routine Access to Oracle RDBMS

A set of files has been added to the SQL$SAMPLE directory which demonstrate the use of Rdb SQL external functions and procedures to access an Oracle RDBMS database. It includes PRO*C source code and build procedures along with an Rdb SQL script to define the external routines. The demonstration is composed of the following files:

- PRO_C_EXT_FUNC.COM
- PRO_C_EXT_FUNC.OPT
- PRO_C_EXT_FUNC.PC
- PRO_C_EXT_FUNC.SQL

The following interactive session shows how to build the shared executable and define the functions in a PERSONNEL database in an environment where ORAUSER.COM has been executed:

```
$ set default MY_DEMO_DIR
$ define PROCEXTFUNC MY_DEMO_DIR
$ copy sql$sample:PRO_C_EXT_FUNC.* *.*
$ @pro_c_ext_func.com

Pro*C/C++: Release 9.2.0.4.0 − Production on Fri May 11 19:34:32 2007

Copyright (c) 1982, 2002, Oracle Corporation.  All rights reserved.

System default option values taken from: ora_proc20:pcscfg.cfg

 − Linking PRO_C_EXT_FUNC.EXE
$ SQL$
SQL> at 'f personnel';
SQL> @PRO_C_EXT_FUNC.SQL
SQL> commit;
SQL> exit;
$
```

The demonstration routines are designed to access the "EMP" table in the "SCOTT" schema in an Oracle RDBMS database. They allow data for this table to be retrieved, both with a singleton retrieval and using a cursor; to be updated; and to be inserted.

The demonstration creates an Rdb stored module named ORA_PRO_C_DEMO_FUNCS that contains the following external routines:

- roif_connect − a function
- roif_disconnect − a function
- roif_commit − a function
- roif_rollback − a function
- roif_get_errmsg − a procedure
- roif_get_employee − a procedure
- roif_open_emp_cursor − a function
- roif_fetch_emp_cursor − a procedure
- roif_close_emp_cursor − a function
- roif_update_employee − a procedure
- roif_insert_employee − a procedure

Chapter 6
Enhancements Provided in Oracle Rdb Release 7.1.5

6.1 Enhancements Provided in Oracle Rdb Release 7.1.5

6.1.1 Enhanced System Table Lookup in Multischema Databases

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

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

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

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

The same query in an older version would fail.

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

This problem has been corrected in Oracle Rdb Release 7.1.5.

6.1.2 Oracle Rdb Release 7.1.x.x New Features Document Name Changed

The file name of the Rdb Release 7.1.x.x New Features document has been changed from NEWFEATURES_71xx to RDB_NEWFEATURES_71xx in order to follow standard naming conventions.

This document is included in saveset A of the Rdb kit and is available in postscript, text and PDF format. This document provides customers with one document to reference to find out about all new features that have been added to the Rdb 7.1 releases.

Chapter 7
Documentation Corrections, Additions and Changes

This chapter provides corrections for documentation errors and omissions.

7.1 Documentation Corrections

7.1.1 Online Backup Can Be Performed With Transfer Via Memory

The following incorrect Oracle RMU BACKUP command restriction will be removed from the Oracle RMU Reference Manual.

In prior releases of the Oracle RMU Reference Manual, it states under the RMU Backup Online option that "However, an online backup operation cannot be performed if TRANSFER VIA MEMORY, also referred to as optimized page transfer, is enabled. (See the description of the SQL ALTER DATABASE statement in the Oracle Rdb SQL Reference Manual for information on optimized page transfer.)". This restriction is no longer true and will be removed from the Oracle RMU Reference Manual.

This restriction is also listed for the Online Copy Database command and for the Online Move Area Command. This restriction is no longer true for these commands either and will be removed from the Oracle RMU Reference Manual.

7.1.2 Missing Example for CREATE STORAGE MAP

Bug 5655348

The SQL Reference Manual did not include an example showing the storage area attributes for a LIST storage map. The following example will appear in a future version of the Oracle Rdb V7.2 SQL Reference Manual in the CREATE STORAGE MAP section.

Example

The following example shows the use of storage area attributes in a LIST storage map. The storage area attributes must be immediately following the storage area name (as in table storage maps).

```
SQL> create database
cont>     filename 'DB$:MULTIMEDIA'
cont>
cont>     create storage area PHOTO_AREA1
cont>         filename 'DB$:PHOTO_AREA1'
cont>         page format UNIFORM
cont>
cont>     create storage area PHOTO_AREA2
cont>         filename 'DB$:PHOTO_AREA2'
cont>         page format UNIFORM
cont>
cont>     create storage area TEXT_AREA
cont>         filename 'DB$:TEXT_AREA'
cont>         page format UNIFORM
cont>
cont>     create storage area AUDIO_AREA
cont>         filename 'DB$:AUDIO_AREA'
cont>         page format UNIFORM
cont>
cont>     create storage area DATA_AREA
```

```
cont>         filename 'DB$:DATA_AREA'
cont>         page format UNIFORM
cont> ;
SQL>
SQL> create table EMPLOYEES
cont>     (name       char(30),
cont>      dob        date,
cont>      ident      integer,
cont>      photograph list of byte varying (4096) as binary,
cont>      resume     list of byte varying (132) as text,
cont>      review     list of byte varying (80) as text,
cont>      voiceprint list of byte varying (4096) as binary
cont>     );
SQL>
SQL> create storage map EMPLOYEES_MAP
cont>     for EMPLOYEES
cont>     enable compression
cont>     store in DATA_AREA;f
SQL>
SQL> create storage map LISTS_MAP
cont>     store lists
cont>         in AUDIO_AREA
cont>                 (thresholds are (89, 99, 100)
cont>                 ,comment is 'The voice clips'
cont>                 ,partition AUDIO_STUFF)
cont>             for (employees.voiceprint)
cont>         in TEXT_AREA
cont>                 (thresholds is (99)
cont>                 ,partition TEXT_DOCUMENTS)
cont>             for (employees.resume, employees.review)
cont>         in (PHOTO_AREA1
cont>                 (comment is 'Happy Smiling Faces?'
cont>                 ,threshold is (99)
cont>                 ,partition PHOTOGRAPHIC_IMAGES_1)
cont>             ,PHOTO_AREA2
cont>                 (comment is 'Happy Smiling Faces?'
cont>                 ,threshold is (99)
cont>                 ,partition PHOTOGRAPHIC_IMAGES_2)
cont>             )
cont>             for (employees.photograph)
cont>             fill randomly
cont>         in RDB$SYSTEM
cont>                 (partition SYSTEM_LARGE_OBJECTS);
SQL>
SQL> show storage map LISTS_MAP;
     LISTS_MAP
 For Lists
 Store clause:          STORE lists
        in AUDIO_AREA
                (thresholds are (89, 99, 100)
                ,comment is 'The voice clips'
                ,partition AUDIO_STUFF)
            for (employees.voiceprint)
        in TEXT_AREA
                (thresholds is (99)
                ,partition TEXT_DOCUMENTS)
            for (employees.resume, employees.review)
        in (PHOTO_AREA1
                (comment is 'Happy Smiling Faces?'
                ,threshold is (99)
                ,partition PHOTOGRAPHIC_IMAGES_1)
            ,PHOTO_AREA2
```

```
                (comment is 'Happy Smiling Faces?'
                ,threshold is (99)
                ,partition PHOTOGRAPHIC_IMAGES_2)
            )
            for (employees.photograph)
            fill randomly
        in RDB$SYSTEM
                (partition SYSTEM_LARGE_OBJECTS)

 Partition information for lists map:
 Vertical Partition: VRP_P000
  Partition: (1) AUDIO_STUFF
    Fill Randomly
   Storage Area: AUDIO_AREA
         Thresholds are (89, 99, 100)
 Comment:       The voice clips
  Partition: (2) TEXT_DOCUMENTS
    Fill Randomly
   Storage Area: TEXT_AREA
         Thresholds are (99, 100, 100)
  Partition: (3) PHOTOGRAPHIC_IMAGES_1
    Fill Randomly
   Storage Area: PHOTO_AREA1
         Thresholds are (99, 100, 100)
 Comment:       Happy Smiling Faces?
  Partition: (3) PHOTOGRAPHIC_IMAGES_2
   Storage Area: PHOTO_AREA2
         Thresholds are (99, 100, 100)
 Comment:       Happy Smiling Faces?
  Partition: (4) SYSTEM_LARGE_OBJECTS
    Fill Randomly
   Storage Area: RDB$SYSTEM
SQL>
SQL> commit;
```

7.1.3 Export Statement: Additional Usage Note

The following usage note will be added to the EXPORT Statement section of the Oracle Rdb SQL Reference Manual the next time it is updated.

Usage Note

The EXPORT statement extracts all metadata and data from the source database in a single transaction. It executes the equivalent to the START DEFAULT TRANSACTION statement.

For example, if you define the RDMS$SET_FLAGS logical name to the TRANSACTION keyword, we can see this single transaction start and commit.

```
$ DEFINE/USER_MODE RDMS$SET_FLAGS TRANSACTION
$ SQL$
SQL> EXPORT DATABASE FILENAME PERSONNEL INTO SAVED_PERSONNEL.RBR;
~T Compile transaction (1) on db: 1
~T Transaction Parameter Block: (len=0)
~T Start_transaction (1) on db: 1, db count=1
~T Commit_transaction (1) on db: 1
~T Prepare_transaction (1) on db: 1
SQL>
```

The *Transaction Parameter Block* of zero length indicates that START DEFAULT TRANSACTION has been executed. The Oracle Rdb server will attempt to define a default definition in the database for the current user and if none is found, a READ ONLY transaction will be started. That is the case in this example.

In some environments, this type of transaction might not be desired. For instance, in an environment with SNAPSHOTS defined as ENABLED DEFERRED, this transaction type would force writers to the database to also start writing to the SNAPSHOT files.

In this case, you can define a PROFILE for the user performing the EXPORT statement and associate a PROFILE with a more appropriate default transaction definition. In this example, we use ISOLATION LEVEL READ COMMITTED to improve the concurrency between EXPORT and other database users.

```
SQL> CREATE PROFILE DB_ADMIN
cont> DEFAULT TRANSACTION
cont>   READ WRITE
cont>   WAIT 10
cont>   ISOLATION LEVEL READ COMMITTED;
SQL> CREATE USER PHILIP IDENTIFIED EXTERNALLY PROFILE DB_ADMIN;
SQL> COMMIT;
```

Now when the EXPORT statement is executed by this user, the default transaction from the profile is used.

```
$ SQL$
SQL> EXPORT DATABASE FILENAME PERSONNEL INTO SAVED_PERSONNEL.RBR;
~T Compile transaction (1) on db: 1
~T Transaction Parameter Block: (len=6)
0000 (00000) TPB$K_VERSION = 1
0001 (00001) TPB$K_ISOLATION_LEVEL1 (read committed)
0002 (00002) TPB$K_WAIT_INTERVAL 10 seconds
0005 (00005) TPB$K_WRITE (read write)
~T Start_transaction (1) on db: 1, db count=1
~T Commit_transaction (1) on db: 1
~T Prepare_transaction (1) on db: 1
SQL>
```

The association with this default transaction can be removed after the EXPORT statement has completed.

```
SQL> ALTER USER PHILIP NO PROFILE;
SQL> COMMIT;
```

7.1.4 RDM$BIND_MAX_DBR_COUNT Documentation Clarification

Bugs 1495227 and 3916606

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) for each database.

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 for each database 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.

Per−Database Value

The RDM$BIND_MAX_DBR_COUNT logical name specifies the maximum number of database recovery processes to run at once for each database. For example, if there are 10 databases being recovered and the value for the RDM$BIND_MAX_DBR_COUNT logical name is 8, up to 80 database recovery processes would be started by the monitor after a node failure.

The RDM$BIND_MAX_DBR_COUNT logical name is translated when the monitor process opens a database. Databases should be closed and reopened for a new value of the logical to become effective.

7.1.5 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 7−1.

Table 7−1 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.

7.1.6 Explanation of SQL$INT in a SQL Multiversion Environment and How to Redefine SQL$INT

Bug 2500495

In an environment running multiple versions of Oracle Rdb, for instance Rdb V7.0 and Rdb V7.1, there are now several varianted SQL images, such as SQL$70.EXE and SQL$71.EXE. However, SQL$INT.EXE is not varianted but acts as a dispatcher using the translation of the logical name RDMS$VERSION_VARIANT to activate the correct SQL runtime environment. This image is replaced when a higher version of Oracle Rdb is installed. Thus, using the example above, when Rdb V7.1 is installed, SQL$INT.EXE will be replaced with the V7.1 SQL$INT.EXE.

If an application is linked in this environment (using V7.1 SQL$INT) and the corresponding executable deployed to a system running Oracle Rdb V7.0 multiversion only, the execution of the application may result in the following error:

```
%IMGACT−F−SYMVECMIS, shareable image's symbol vector table mismatch
```

In order to avoid such a problem, the following alternative is suggested:

In the multiversion environment running both Oracle Rdb V7.0 and Oracle Rdb V7.1, run Oracle Rdb V7.0 multiversion by running the command procedures RDB$SETVER.COM 70 and RDB$SETVER RESET. This will set up the necessary logical names and symbols that establish the Oracle Rdb V7.0 environment.

For example:

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

Current PROCESS Oracle Rdb environment is version V7.0−63 (MULTIVERSION)
Current PROCESS SQL environment is version V7.0−63 (MULTIVERSION)
Current PROCESS Rdb/Dispatch environment is version V7.0−63 (MULTIVERSION)

$ @SYS$LIBRARY:RDB$SERVER RESET
```

Now run SQL and verify that the version is correct:

```
$ sql$
SQL> show version
Current version of SQL is: Oracle Rdb SQL V7.0−63
```

Define SQL$INT to point to the varianted SQL$SHR.EXE. Then, create an options file directing the linker to link with this newly defined SQL$INT. An example follows:

```
$ DEFINE SQL$INT SYS$SHARE:SQL$SHR'RDMS$VERSION_VARIANT'.EXE
$ LINK TEST_APPL,SQL$USER/LIB,SYS$INPUT/option
SQL$INT/SHARE
^Z
```

The executable is now ready to be deployed to the Oracle Rdb V7.0 multiversion environment and should run successfully.

Please note that with each release of Oracle Rdb, new entry points are added to the SQL$INT shareable image. This allows the implementation of new functionality. Therefore, applications linked with SQL$INT from Oracle Rdb V7.1 cannot be run on systems with only Oracle Rdb V7.0 installed. This is because the shareable image does not contain sufficient entry points.

The workaround presented here allows an application to explicitly link with the Oracle Rdb V7.0 version of the image. Such applications are upward compatible and will run on Oracle Rdb V7.0 and Oracle Rdb V7.1. The applications should be compiled and linked under the lowest version.

In environments where Oracle Rdb V7.1 is installed, this workaround is not required because the SQL$INT image will dynamically activate the appropriate SQL$SHRxx image as expected.

7.1.7 Documentation Omitted Several Reserved Words

Bug 2319321

The following keywords are considered reserved words in Oracle Rdb Release 7.1.

- UID
- CURRENT_UID
- SYSTEM_UID
- SESSION_UID
- RAW
- LONG
- DBKEY
- ROWID
- SYSDATE

In particular, any column which has these names will be occluded by the keyword. i.e. selecting from column UID will be interpreted as referencing the built in function UID and so return a different result.

The correction to this problem is to enable keyword quoting using SET QUOTING RULES 'SQL92' (or 'SQL99') and enclose the column name in quotations.

In addition, SQL will now generate a warning if these reserved words are used (unquoted) in CREATE and ALTER operations.

7.1.8 Using Databases from Releases Earlier Than V6.0

Bug 2383967

You cannot convert or restore databases earlier than V6.0 directly to V7.1. The RMU Convert command for V7.1 supports conversions from V6.0 through V7.0 only. If you have a V3.0 through V5.1 database, you must convert it to at least V6.0 and then convert it to V7.1. For example, if you have a V4.2 database, convert it first to at least V6.0, then convert the resulting database to V7.1.

If you attempt to convert a database created prior to V6.0 directly to V7.1, Oracle RMU generates an error.

7.1.9 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.

7.1.10 New Request Options for RDO, RDBPRE and RDB$INTERPRET

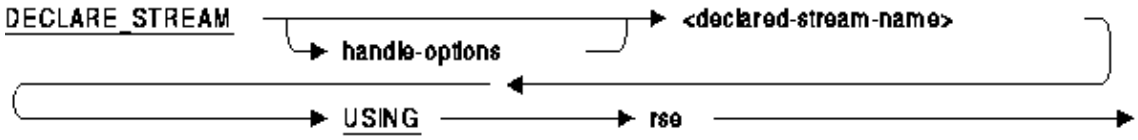
This release note was included in the V70A Release Notes but had gotten dropped somewhere along the line.

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.

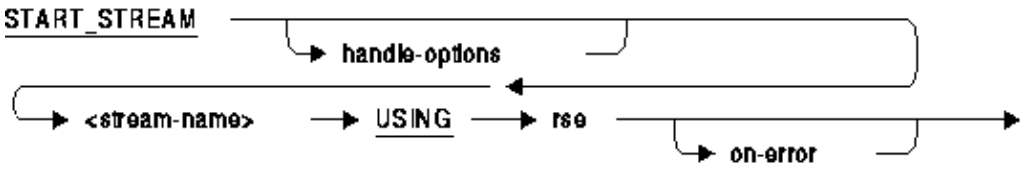
In prior releases, the handle−options could not be specified in interactive RDO or RDB$INTERPRET. This has changed in Rdb7 but these allowed options will be limited to MODIFY and PROTECTED keywords. For RDBPRE, all options listed will be supported. These option names were chosen to be existing keywords 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.

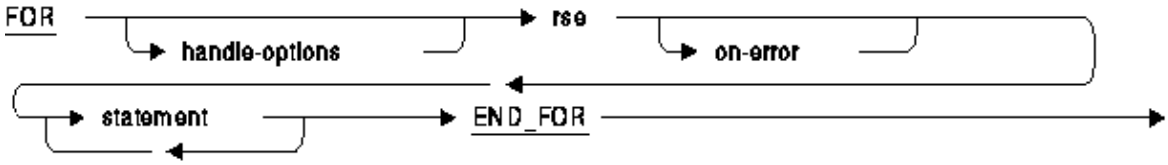
Example 5−1 DECLARE_STREAM Format



Example 5−2 START_STREAM Format

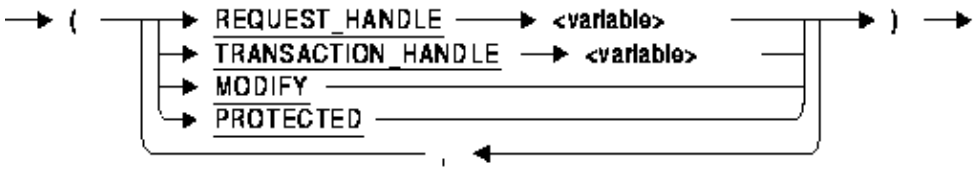


Example 5−3 FOR Format



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.

7.2 Address and Phone Number Correction for Documentation

In release 7.0 or earlier documentation, the address and fax phone number listed on the Send Us Your Comments page are incorrect. The correct information is:

```
FAX -- 603.897.3825
Oracle Corporation
One Oracle Drive
Nashua, NH 03062-2804
USA
```

7.3 Online Document Format

You can view the documentation in Adobe Acrobat format using the Acrobat Reader, which allows anyone to view, navigate, and print documents in the Adobe Portable Document Format (PDF). See http://www.adobe.com for information about obtaining a free copy of Acrobat Reader and for information on supported platforms.

The Oracle Rdb documentation in Adobe Acrobat format is available on MetaLink:

```
Top Tech Docs\Oracle Rdb\Documentation\<bookname>
```

7.4 New and Changed Features in Oracle Rdb Release 7.1

This section provides information about late−breaking new features or information that is missing or changed since the Oracle Rdb New and Changed Features for Oracle Rdb manual was published.

7.5 Oracle Rdb7 and Oracle CODASYL DBMS Guide to Hot Standby Databases

This section provides information that is missing from or changed in V7.0 of the Oracle Rdb7 and Oracle CODASYL DBMS Guide to Hot Standby Databases.

7.5.1 Restrictions Lifted on After−Image Journal Files

The Hot Standby software has been enhanced regarding how it handles after−image journal files. Section 4.2.4 in the Oracle Rdb and Oracle CODASYL DBMS Guide to Hot Standby Databases states the following information:

```
If an after−image journal switchover operation is suspended when
replication operations are occurring, you must back up one or more of
the modified after−image journals to add a new journal file.
```

This restriction has been removed. Now, you can add journal files or use the emergency AIJ feature of Oracle Rdb release 7.0 to automatically add a new journal file. Note the following distinctions between adding an AIJ file and adding an emergency AIJ file:

- You can add an AIJ file to the master database and it will be replicated on the standby database. If replication operations are active, the AIJ file is created on the standby database immediately. If replication operations are not active, the AIJ file is created on the standby database when replication operations are restarted.
- You can add emergency AIJ files anytime. If replication operations are active, the emergency AIJ file is created on the standby database immediately. However, because emergency AIJ files are not journaled, starting replication after you create an emergency AIJ will fail. You cannot start replication operations because the Hot Standby software detects a mismatch in the number of after−image journal files on the master compared to the standby database.
  If an emergency AIJ file is created on the master database when replication operations are not active, you must perform a master database backup and then restore the backup on the standby database. Otherwise, an AIJSIGNATURE error results.

7.5.2 Changes to RMU Replicate After_Journal ... Buffer Command

The behavior of the RMU Replicate After_Journal ... 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 will use a minimum of 4096 buffers. The value provided to the Buffers qualifier will be accepted but 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, then 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, then 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 qualifier is specified, then 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 then 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.

7.5.3 Unnecessary Command in the Hot Standby Documentation

There is an unnecessary command documented in the Oracle Rdb and Oracle CODASYL DBMS Guide to Hot Standby Databases manual. The documentation (in Section 2.12 "Step 10: Specify the Network Transport Protocol") says that to use TCP/IP as the network protocol, you must issue the following commands:

```
$ CONFIG UCX AIJSERVER OBJECT
$ UCX SET SERVICE RDMAIJSRV
/PORT=n
/USER_NAME=RDMAIJSERVER
/PROCESS_NAME=RDMAIJSERVER
/FILE=SYS$SYSTEM:rdmaijserver_ucx.com
/LIMIT=nn
```

The first of these commands ($ CONFIG UCX AIJSERVER OBJECT) is unnecessary. You can safely disregard the first line when setting up to use TCP/IP with Hot Standby.

The documentation will be corrected in a future release of Oracle Rdb.

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

Starting with Oracle Rdb Release 7.0.2.1, the RDMAIJ image became 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 with regard to setting up the RDMAIJSERVER object when using UCX as the network transport protocol. The UCX SET SERVICE command is now similar to the following:

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

For Oracle Rdb multiversion, the UCX SET SERVICE command is 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. 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 varianted image does not impact installations using DECNet since the correct DECNet object is created during the Oracle Rdb installation.

7.5.5 CREATE INDEX Operation Supported for Hot Standby

On Page 1−13 of the Oracle Rdb7 and Oracle CODASYL DBMS 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.

7.6 Oracle Rdb7 for OpenVMS Installation and Configuration Guide

This section provides information that is missing from or changed in V7.0 of the Oracle Rdb7 for OpenVMS Installation and Configuration Guide.

7.6.1 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". This section 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 Oracle Rdb images with the Resident qualifier.

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

Oracle Corporation recommends 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 GH_RSRVPGCNT to be a value of zero in order to ensure the highest level of system performance.

7.6.2 Prerequisite Software

In addition to the software listed in the Oracle Rdb Installation and Configuration Guide and at the url http://www.oracle.com/rdb/product_info/index.html, note that the MACRO−32 compiler and the OpenVMS linker are required OpenVMS components in order to install Oracle Rdb on your OpenVMS Alpha system.

The MACRO−32 Compiler for OpenVMS Alpha is a standard component of the OpenVMS Operating System. It is used to compile VAX MACRO assembly language source files into native OpenVMS Alpha object code. During the Oracle Rdb installation procedure, and portions of the installation verification procedure (such as the test for RDBPRE), the MACRO−32 compiler is required.

The OpenVMS linker is a standard component of the OpenVMS Operating System. It is used to link one or more input files into a program image and defines the execution characteristics of the image. The linker will be required for application development and is likewise used by the Oracle Rdb installation procedure and the installation verification procedure.

7.6.3 Defining the RDBSERVER Logical Name

Sections 4.3.7.1 and 4.3.7.2 in the Oracle Rdb7 for OpenVMS Installation and Configuration Guide provide the following examples for defining the RDBSERVER logical name: *$ DEFINE RDBSERVER SYS$SYSTEM:RDBSERVER70.EXE*

and *$ DEFINE RDBSERVER SYS$SYSTEM:RDBSERVER61.EXE*

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

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

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

The correct definition of the logical name is as follows: *DEFINE RDBSERVER SYS$COMMON:<SYSEXE>RDBSERVER70.EXE*

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

7.7 Guide to Database Design and Definition

This section provides information that is missing from or changed in release 7.0 of the Oracle Rdb7 Guide to Database Design and Definition.

7.7.1 Lock Timeout Interval Logical Incorrect

On Page 7−31 of Section 7.4.8 in the Oracle Rdb7 Guide to Database Design and Definition, the RDM$BIND_LOCK_TIMEOUT logical name is referenced incorrectly. The correct logical name is RDM$BIND_LOCK_TIMEOUT_INTERVAL.

The Oracle Rdb7 Guide to Database Design and Definition will be corrected in a future release.

7.7.2 Example 4−13 and Example 4−14 Are Incorrect

Example 4−13 showing vertical partitioning, and Example 4−14, showing vertical and horizontal partitioning, are incorrect. They should appear as follows:

Example 4−13:

```
SQL> CREATE STORAGE MAP EMPLOYEES_1_MAP
cont>    FOR EMPLOYEES
cont>    ENABLE COMPRESSION
cont>    STORE COLUMNS (EMPLOYEE_ID, LAST_NAME, FIRST_NAME,
cont>                   MIDDLE_INITIAL, STATUS_CODE)
cont>                   DISABLE COMPRESSION
cont>                   IN ACTIVE_AREA
cont>    STORE COLUMNS (ADDRESS_DATA_1, ADDRESS_DATA_2, CITY,
cont>                   STATE, POSTAL_CODE)
cont>                   IN INACTIVE_AREA
cont>    STORE IN OTHER_AREA;
```

Example 4−14:

```
SQL>  CREATE STORAGE MAP EMPLOYEES_1_MAP2
cont>        FOR EMP2
cont>        STORE COLUMNS (EMPLOYEE_ID, LAST_NAME, FIRST_NAME,
cont>                       MIDDLE_INITIAL, STATUS_CODE)
cont>              USING (EMPLOYEE_ID)
cont>              IN ACTIVE_AREA_A WITH LIMIT OF ('00399')
cont>              IN ACTIVE_AREA_B WITH LIMIT OF ('00699')
cont>              OTHERWISE IN ACTIVE_AREA_C
cont>        STORE COLUMNS (ADDRESS_DATA_1, ADDRESS_DATA_2, CITY,
cont>                      STATE, POSTAL_CODE)
cont>              USING (EMPLOYEE_ID)
cont>              IN INACTIVE_AREA_A WITH LIMIT OF ('00399')
cont>              IN INACTIVE_AREA_B WITH LIMIT OF ('00699')
cont>              OTHERWISE IN INACTIVE_AREA_C
cont>        STORE IN OTHER_AREA;
```

7.8 Oracle RMU Reference Manual, Release 7.0

This section provides information that is missing from or changed in V7.0 of the Oracle RMU Reference Manual.

7.8.1 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 higher 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
```

7.8.2 New Transaction_Mode Qualifier for Oracle RMU Commands

A new qualifier, Transaction_Mode, has been added to the RMU Copy, Move_Area, Restore, and Restore Only_Root commands. You can use this qualifier to set the allowable transaction modes for the database root file created by these commands. If you are not creating a root file as part of one of these commands, for example, you are restoring an area, attempting to use this qualifier returns a CONFLSWIT error. This qualifier is similar to the SET TRANSACTION MODE clause of the CREATE DATABASE command in interactive SQL.

The primary use of this qualifier is when you restore a backup file (of the master database) to create a Hot Standby database. Include the Transaction_Mode qualifier on the RMU Restore command when you create the standby database (prior to starting replication operations). Because only read−only transactions are allowed on the standby database, you should use the Transaction_Mode=Read_Only qualifier setting. This setting prevents modifications to the standby database at all times, even when replication operations are not active.

You can specify the following transaction modes for the Transaction_Mode qualifier:

```
All
Current
None
[No]Batch_Update
[No]Read_Only
[No]Exclusive
[No]Exclusive_Read
[No]Exclusive_Write
[No]Protected
[No]Protected_Read
[No]Protected_Write
[No]Shared
[No]Shared_Read
[No]Shared_Write
```

Note that [No] indicates that the value can be negated. For example, the NoExclusive_Write option indicates that exclusive write is not an allowable access mode for this database. If you specify the Shared, Exclusive, or Protected option, Oracle RMU assumes you are referring to both reading and writing in these modes. For example, the Transaction_Mode=Shared option indicates that you want both Shared_Read and Shared_Write as transaction modes. No mode is enabled unless you add that mode to the list or you use the ALL option to enable all modes.

You cannot negate the following three options: All, which enables all transaction modes; None, which disables all transaction modes; and Current, which enables all transaction modes that are set for the source database. If you do not specify the Transaction_Mode qualifier, Oracle RMU uses the transaction modes enabled for the source database.

You can list one qualifier that enables or disables a particular mode followed by another that does the opposite. For example, Transaction_Mode=(NoShared_Write, Shared) is ambiguous because the first value disables Shared_Write access while the second value enables Shared_Write access. Oracle RMU resolves the ambiguities by first enabling all modes that are enabled by the items in the Transaction_Mode list and then disabling those modes that are disabled by items in the Transaction_Mode list. The order of items in the list is irrelevant. In the example discussed, Shared_Read is enabled and Shared_Write is disabled.

The following example shows how to set a newly restored database to allow read−only transactions only. After Oracle RMU executes the command, the database is ready for you to start Hot Standby replication operations.

```
$ RMU/RESTORE/TRANSACTION_MODE=READ_ONLY MF_PERSONNEL.RBF
```

7.8.3 RMU Server After_Journal Stop Command

If database replication is active and you attempt to stop the database AIJ Log Server, Oracle Rdb returns an error. You must stop database replication before attempting to stop the server.

In addition, a new qualifier, Output=filename, has been added to the RMU Server After_Journal Stop command. This optional qualifier allows you to specify the file where the operational log is to be created. The operational log records the transmission and receipt of network messages.

If you do not include a directory specification with the file name, the log file is created in the database root file directory. It is invalid to include a node name as part of the file name specification.

Note that all Hot Standby bugcheck dumps are written to the corresponding bugcheck dump file; bugcheck dumps are not written to the file you specify with the Output qualifier.

7.8.4 Incomplete Description of Protection Qualifier for RMU Backup After_Journal Command

The description of the Protection Qualifier for the RMU Backup After_Journal command is incomplete in the Oracle RMU Reference Manual for Digital UNIX. The complete description is as follows:

The Protection qualifier specifies the system file protection for the backup file produced by the RMU Backup After_Journal command. If you do not specify the Protection qualifier, the default access permissions are −rw−r−−−−− for backups to disk or tape.

Tapes do not allow delete or execute access and the superuser account always has both read and write access to tapes. In addition, a more restrictive class accumulates the access rights of the less restrictive classes.

If you specify the Protection qualifier explicitly, the differences in access permissions applied for backups to tape or disk as noted in the preceding paragraph are applied. Thus, if you specify Protection=(S,O,G:W,W:R), the access permissions on tape becomes rw−rw−r−.

7.8.5 RMU Extract Command Options Qualifier

A documentation error exists in the description of the Options=options−list qualifier of the RMU Extract command. Currently, the documentation states that this qualifier is not applied to output created by the Items=Volume qualifier. This is incorrect. Beginning with 6.1 of Oracle Rdb, the behavior of the Options=options−list qualifier is applied to output created by the Items=Volume qualifier.

7.8.6 RDM$SNAP_QUIET_POINT Logical is Incorrect

On page 2−72 of the Oracle RMU Reference Manual, the reference to the RDM$SNAP_QUIET_POINT logical is incorrect. The correct logical name is RDM$BIND_SNAP_QUIET_POINT.

7.8.7 Using Delta Time with RMU Show Statistics Command

Oracle RMU does not support the use of delta time. However, because the OpenVMS platform does, there is a

workaround. You can specify delta time using the following syntax with the RMU Show Statistics command:

```
$ RMU/SHOW STATISTICS/OUTPUT=file-spec/UNTIL=" ' ' f$cvtime ("+7:00") ' "
```

The +7:00 adds 7 hours to the current time.

You can also use "TOMORROW" and "TODAY+n".

This information will be added to the description of the Until qualifier of the RMU Show Statistics command in a future release of the Oracle RMU Reference Manual.

7.9 Oracle Rdb7 Guide to Database Performance and Tuning

The following section provides corrected, clarified, or omitted information for the Oracle Rdb7 Guide to Database Performance and Tuning manual.

7.9.1 Dynamic OR Optimization Formats

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].

7.9.2 Oracle Rdb Logical Names

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

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.

7.9.3 Waiting for Client Lock Message

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

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

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

The lock name for these special locks consist of an encoded 16 byte name. The first 4 bytes contains the leading four bytes of the user name (for system objects the RDB$ prefix is skipped) followed by three longwords. The lock is displayed in text format first − here will be seen the prefix for the table, routine, or module name; followed by its hexadecimal representation. The text version masks out non−printable characters with a dot (.).

```
waiting for client '...."...EMPL' 4C504D450000002200000004000000055
```

The leftmost value seen in the hexadecimal output contains the name prefix which is easier read in the text field. Then comes a hex number (00000022) which is the id of the object. The id is described below for tables, views, functions, procedures, modules, and sequences.

- 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 (that is functions and procedures), 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.
- For sequences, the id represents the unique value found in the RDB$SEQUENCE_ID column of the RDB$SEQUENCES system relation for the given sequence.

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

Table 7−2 Objects and Their Hexadecimal Type Value

Object	Hexadecimal Value
Tables or views	00000004
Modules	00000015
Routines	00000016
Sequences	00000019

The last value in the hexadecimal output represents the lock type. The hexadecimal value 55 indicates this is a client lock and distinct from page and other data structure locks.

The following example shows a "waiting for client lock" message from a Stall Messages screen while the application was processing the EMPLOYEES table from MF_PERSONNEL. The terminal should be set to 132 characters wide to view the full client lock string.

```
Process.ID  Since.................. T Stall.reason.............................Lock.ID.
27800643:1                           waiting for logical area 79 (CW)         16004833
27800507:1  31−OCT−2002 16:05:15.71 W waiting for client '...."...EMPL' 4C504D450000002200000004
```

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$ROUTINE_ID = 7;
SQL> select RDB$SEQUENCE_NAME from RDB$SEQUENCES where RDB$SEQUENCE_ID = 2;
```

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.

7.9.4 RDMS$TTB_HASH_SIZE Logical Name

The logical name RDMS$TTB_HASH_SIZE sets the size of the hash table used for temporary tables. If the logical name is not defined, Oracle Rdb uses a default value of 1249.

If you expect that temporary tables will be large (that is, 10K or more rows), use this logical name to adjust the hash table size to avoid long hash chains. Set the value to approximately 1/4 of the expected maximum number of rows for each temporary table. For example, if a temporary table will be populated with 100,000 rows, define this logical name to be 25000. If there are memory constraints on your system, you should define the logical name to be no higher than this value (1/4 of the expected maximum number of rows).

7.9.5 Error in Updating and Retrieving a Row by Dbkey Example 3−22

Example 3−22 in Section 3.8.3 that shows how to update and retrieve a row by dbkey is incorrect. The example should appear as follows:

```
SQL> ATTACH 'FILENAME MF_PERSONNEL.RDB';
SQL> --
SQL> -- Declare host variables
SQL> --
SQL> DECLARE :hv_row INTEGER;              -- Row counter
SQL> DECLARE :hv_employee_id ID_DOM;       -- EMPLOYEE_ID field
SQL> DECLARE :hv_employee_id_ind SMALLINT; -- Null indicator variable
SQL> --
SQL> DECLARE :hv_dbkey CHAR(8);            -- DBKEY storage
SQL> DECLARE :hv_dbkey_ind SMALLINT;       -- Null indicator variable
SQL> --
SQL> DECLARE :hv_last_name LAST_NAME_DOM;
SQL> DECLARE :hv_new_address_data_1 ADDRESS_DATA_1_DOM;
SQL> --
SQL> SET TRANSACTION READ WRITE;
SQL> BEGIN
cont> --
cont> -- Set the search value for SELECT
cont> --
cont>  SET :hv_last_name = 'Ames';
cont> --
cont> -- Set the NEW_ADDRESS_DATA_1 value
cont> --
cont> SET :hv_new_address_data_1 = '100 Broadway Ave.';
cont> END;
SQL> COMMIT;
SQL> --
SQL> SET TRANSACTION READ ONLY;
SQL> BEGIN
cont> SELECT E.EMPLOYEE_ID, E.DBKEY
cont>   INTO :hv_employee_id INDICATOR :hv_employee_id_ind,
cont>        :hv_dbkey INDICATOR :hv_dbkey_ind
cont>   FROM EMPLOYEES E
cont> WHERE E.LAST_NAME = :hv_last_name
cont> LIMIT TO 1 ROW;
cont> --
cont> GET DIAGNOSTICS :hv_row = ROW_COUNT;
cont> END;
SQL> COMMIT;
```

```
SQL> --
SQL> SET TRANSACTION READ WRITE RESERVING EMPLOYEES FOR SHARED WRITE;
SQL> BEGIN
cont> IF (:hv_row = 1) THEN
cont>    BEGIN
cont>    UPDATE EMPLOYEES E
cont>      SET E.ADDRESS_DATA_1 = :hv_new_address_data_1
cont>    WHERE E.DBKEY = :hv_dbkey;
cont>    END;
cont> END IF;
cont> END;
SQL> COMMIT;
SQL> --
SQL> -- Display result of change
SQL> --
SQL> SET TRANSACTION READ ONLY;
SQL> SELECT E.*
cont> FROM EMPLOYEES E
cont> WHERE E.DBKEY = :hv_dbkey;
 EMPLOYEE_ID   LAST_NAME        FIRST_NAME   MIDDLE_INITIAL
   ADDRESS_DATA_1              ADDRESS_DATA_2         CITY
      STATE   POSTAL_CODE   SEX    BIRTHDAY      STATUS_CODE
 00416         Ames             Louie        A
   100 Broadway Ave.                                  Alton
      NH      03809         M      13-Apr-1941   1

1 row selected
SQL>
```

The new example will appear in a future publication of the Oracle Rdb7 Guide to Database Performance and Tuning manual.

7.9.6 Error in Calculation of Sorted Index in Example 3−46

Example 3−46 in Section 3.9.5.1 shows the output when you use the RMU Analyze Indexes command and specify the Option=Debug qualifier and the DEPARTMENTS_INDEX sorted index.

The description of the example did not include the 8 byte dbkey in the calculation of the sorted index. The complete description is as follows:

The entire index (26 records) is located on pages 2 and 3 in logical area 72 and uses 188 bytes of a possible 430 bytes or the node record is 47 percent full. Note that due to index compression, the node size has decreased in size from 422 bytes to 188 bytes and the percent fullness of the node records has dropped from 98 to 47 percent. Also note that the used/avail value in the summary information at the end of the output does not include the index header and trailer information, which accounts for 32 bytes. This value is shown for each node record in the detailed part of the output. The number of bytes used by the index is calculated as follows: the sort key is 4 bytes plus a null byte for a total of 5 bytes. The prefix is 1 byte and the suffix is 1 byte. The prefix indicates the number of bytes in the preceding key that are the same and the suffix indicates the number of bytes that are different from the preceding key. The dbkey pointer to the row is 8 bytes. There are 26 data rows multiplied by 15 bytes for a total of 390 bytes. The 15 bytes include:

- 7 bytes for the sort key: length + null byte + prefix + suffix
- 8 bytes for the dbkey pointer to the row

Add 32 bytes for index header and trailer information for the index node to the 390 bytes for a total of 422

bytes used. Index compression reduces the number of bytes used to 188 bytes used.

The revised description will appear in a future publication of the Oracle Rdb7 Guide to Database Performance and Tuning manual.

7.9.7 Documentation Error in Section C.7

The Oracle Rdb Guide to Database Performance And Tuning, Volume 2 contains an error in Section C.7 titled 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, the statistics should be described as show below:

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

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

7.9.8 Missing Tables Descriptions for the RDBEXPERT Collection Class

Appendix B in the Oracle Rdb7 Guide to Database Performance and Tuning describes the event−based data tables in the formatted database for the Oracle Rdb PERFORMANCE and RDBEXPERT collection classes. This section describes the missing tables for the RDBEXPERT collection class.

Table 7−3 shows the TRANS_TPB table.

Table 7−3 Columns for Table EPC$1_221_TRANS_TPB

Column Name	Data Type	Domain
COLLECTION_RECORD_ID	SMALLINT	COLLECTION_RECORD_ID_DOMAIN
IMAGE_RECORD_ID	INTEGER	IMAGE_RECORD_ID_DOMAIN
CONTEXT_NUMBER	INTEGER	CONTEXT_NUMBER_DOMAIN
TIMESTAMP_POINT	DATE VMS	
CLIENT_PC	INTEGER	
STREAM_ID	INTEGER	
TRANS_ID	VARCHAR(16)	
TRANS_ID_STR_ID	INTEGER	STR_ID_DOMAIN
TPB	VARCHAR(127)	
TPB_STR_ID	INTEGER	STR_ID_DOMAIN

Table 7−4 shows the TRANS_TPB_ST table. An index is provided for this table. It is defined with column STR_ID, duplicates are allowed, and the type is sorted.

Table 7−4 Columns for Table EPC$1_221_TRANS_TPB_ST

Column Name	Data Type	Domain
STR_ID	INTEGER	STR_ID_DOMAIN
SEGMENT_NUMBER	SMALLINT	SEGMENT_NUMBER_DOMAIN
STR_SEGMENT	VARCHAR(128)	

7.9.9 Missing Columns Descriptions for Tables in the Formatted Database

Some of the columns were missing from the tables in Appendix B in the Oracle Rdb7 Guide to Database Performance and Tuning. The complete table definitions are described in this section.

Table 7−5 shows the DATABASE table.

Table 7−5 Columns for Table EPC$1_221_DATABASE

Column Name	Data Type	Domain
COLLECTION_RECORD_ID	SMALLINT	COLLECTION_RECORD_ID_DOMAIN
IMAGE_RECORD_ID	INTEGER	IMAGE_RECORD_ID_DOMAIN
CONTEXT_NUMBER	INTEGER	CONTEXT_NUMBER_DOMAIN
TIMESTAMP_POINT	DATE VMS	
CLIENT_PC	INTEGER	
STREAM_ID	INTEGER	
DB_NAME	VARCHAR(255)	
DB_NAME_STR_ID	INTEGER	STR_ID_DOMAIN
IMAGE_FILE_NAME	VARCHAR(255)	
IMAGE_FILE_NAME_STR_ID	INTEGER	STR_ID_DOMAIN

Table 7−6 shows the REQUEST_ACTUAL table.

Table 7−6 Columns for Table EPC$1_221_REQUEST_ACTUAL

Column Name	Data Type	Domain
COLLECTION_RECORD_ID	SMALLINT	COLLECTION_RECORD_ID_DOMAIN
IMAGE_RECORD_ID	INTEGER	IMAGE_RECORD_ID_DOMAIN
CONTEXT_NUMBER	INTEGER	CONTEXT_NUMBER_DOMAIN
TIMESTAMP_START	DATE VMS	

TIMESTAMP_END	DATE VMS
DBS_READS_START	INTEGER
DBS_WRITES_START	INTEGER
RUJ_READS_START	INTEGER
RUJ_WRITES_START	INTEGER
AIJ_WRITES_START	INTEGER
ROOT_READS_START	INTEGER
ROOT_WRITES_START	INTEGER
BUFFER_READS_START	INTEGER
GET_VM_BYTES_START	INTEGER
FREE_VM_BYTES_START	INTEGER
LOCK_REQS_START	INTEGER
REQ_NOT_QUEUED_START	INTEGER
REQ_STALLS_START	INTEGER
REQ_DEADLOCKS_START	INTEGER
PROM_DEADLOCKS_START	INTEGER
LOCK_RELS_START	INTEGER
LOCK_STALL_TIME_START	INTEGER
D_FETCH_RET_START	INTEGER
D_FETCH_UPD_START	INTEGER
D_LB_ALLOK_START	INTEGER
D_LB_GBNEEDLOCK_START	INTEGER
D_LB_NEEDLOCK_START	INTEGER
D_LB_OLDVER_START	INTEGER
D_GB_NEEDLOCK_START	INTEGER
D_GB_OLDVER_START	INTEGER
D_NOTFOUND_IO_START	INTEGER
D_NOTFOUND_SYN_START	INTEGER
S_FETCH_RET_START	INTEGER
S_FETCH_UPD_START	INTEGER
S_LB_ALLOK_START	INTEGER
S_LB_GBNEEDLOCK_START	INTEGER
S_LB_NEEDLOCK_START	INTEGER
S_LB_OLDVER_START	INTEGER
S_GB_NEEDLOCK_START	INTEGER
S_GB_OLDVER_START	INTEGER
S_NOTFOUND_IO_START	INTEGER
S_NOTFOUND_SYN_START	INTEGER
D_ASYNC_FETCH_START	INTEGER
S_ASYNC_FETCH_START	INTEGER
D_ASYNC_READIO_START	INTEGER
S_ASYNC_READIO_START	INTEGER

AS_READ_STALL_START	INTEGER	
AS_BATCH_WRITE_START	INTEGER	
AS_WRITE_STALL_START	INTEGER	
BIO_START	INTEGER	
DIO_START	INTEGER	
PAGEFAULTS_START	INTEGER	
PAGEFAULT_IO_START	INTEGER	
CPU_START	INTEGER	
CURRENT_PRIO_START	SMALLINT	
VIRTUAL_SIZE_START	INTEGER	
WS_SIZE_START	INTEGER	
WS_PRIVATE_START	INTEGER	
WS_GLOBAL_START	INTEGER	
CLIENT_PC_END	INTEGER	
STREAM_ID_END	INTEGER	
REQ_ID_END	INTEGER	
COMP_STATUS_END	INTEGER	
REQUEST_OPER_END	INTEGER	
TRANS_ID_END	VARCHAR(16)	
TRANS_ID_END_STR_ID	INTEGER	STR_ID_DOMAIN
DBS_READS_END	INTEGER	
DBS_WRITES_END	INTEGER	
RUJ_READS_END	INTEGER	
RUJ_WRITES_END	INTEGER	
AIJ_WRITES_END	INTEGER	
ROOT_READS_END	INTEGER	
ROOT_WRITES_END	INTEGER	
BUFFER_READS_END	INTEGER	
GET_VM_BYTES_END	INTEGER	
FREE_VM_BYTES_END	INTEGER	
LOCK_REQS_END	INTEGER	
REQ_NOT_QUEUED_END	INTEGER	
REQ_STALLS_END	INTEGER	
REQ_DEADLOCKS_END	INTEGER	
PROM_DEADLOCKS_END	INTEGER	
LOCK_RELS_END	INTEGER	
LOCK_STALL_TIME_END	INTEGER	
D_FETCH_RET_END	INTEGER	
D_FETCH_UPD_END	INTEGER	
D_LB_ALLOK_END	INTEGER	
D_LB_GBNEEDLOCK_END	INTEGER	
D_LB_NEEDLOCK_END	INTEGER	

D_LB_OLDVER_END	INTEGER
D_GB_NEEDLOCK_END	INTEGER
D_GB_OLDVER_END	INTEGER
D_NOTFOUND_IO_END	INTEGER
D_NOTFOUND_SYN_END	INTEGER
S_FETCH_RET_END	INTEGER
S_FETCH_UPD_END	INTEGER
S_LB_ALLOK_END	INTEGER
S_LB_GBNEEDLOCK_END	INTEGER
S_LB_NEEDLOCK_END	INTEGER
S_LB_OLDVER_END	INTEGER
S_GB_NEEDLOCK_END	INTEGER
S_GB_OLDVER_END	INTEGER
S_NOTFOUND_IO_END	INTEGER
S_NOTFOUND_SYN_END	INTEGER
D_ASYNC_FETCH_END	INTEGER
S_ASYNC_FETCH_END	INTEGER
D_ASYNC_READIO_END	INTEGER
S_ASYNC_READIO_END	INTEGER
AS_READ_STALL_END	INTEGER
AS_BATCH_WRITE_END	INTEGER
AS_WRITE_STALL_END	INTEGER
BIO_END	INTEGER
DIO_END	INTEGER
PAGEFAULTS_END	INTEGER
PAGEFAULT_IO_END	INTEGER
CPU_END	INTEGER
CURRENT_PRIO_END	SMALLINT
VIRTUAL_SIZE_END	INTEGER
WS_SIZE_END	INTEGER
WS_PRIVATE_END	INTEGER
WS_GLOBAL_END	INTEGER

Table 7−7 shows the TRANSACTION table.

Table 7−7 Columns for Table EPC$1_221_TRANSACTION

Column Name	Data Type	Domain
COLLECTION_RECORD_ID	SMALLINT	COLLECTION_RECORD_ID_DOMAIN
IMAGE_RECORD_ID	INTEGER	IMAGE_RECORD_ID_DOMAIN
CONTEXT_NUMBER	INTEGER	CONTEXT_NUMBER_DOMAIN
TIMESTAMP_START	DATE VMS	

TIMESTAMP_END	DATE VMS	
CLIENT_PC_START	INTEGER	
STREAM_ID_START	INTEGER	
LOCK_MODE_START	INTEGER	
TRANS_ID_START	VARCHAR(16)	
TRANS_ID_START_STR_ID	INTEGER	STR_ID_DOMAIN
GLOBAL_TID_START	VARCHAR(16)	
GLOBAL_TID_START_STR_ID	INTEGER	STR_ID_DOMAIN
DBS_READS_START	INTEGER	
DBS_WRITES_START	INTEGER	
RUJ_READS_START	INTEGER	
RUJ_WRITES_START	INTEGER	
AIJ_WRITES_START	INTEGER	
ROOT_READS_START	INTEGER	
ROOT_WRITES_START	INTEGER	
BUFFER_READS_START	INTEGER	
GET_VM_BYTES_START	INTEGER	
FREE_VM_BYTES_START	INTEGER	
LOCK_REQS_START	INTEGER	
REQ_NOT_QUEUED_START	INTEGER	
REQ_STALLS_START	INTEGER	
REQ_DEADLOCKS_START	INTEGER	
PROM_DEADLOCKS_START	INTEGER	
LOCK_RELS_START	INTEGER	
LOCK_STALL_TIME_START	INTEGER	
D_FETCH_RET_START	INTEGER	
D_FETCH_UPD_START	INTEGER	
D_LB_ALLOK_START	INTEGER	
D_LB_GBNEEDLOCK_START	INTEGER	
D_LB_NEEDLOCK_START	INTEGER	
D_LB_OLDVER_START	INTEGER	
D_GB_NEEDLOCK_START	INTEGER	
D_GB_OLDVER_START	INTEGER	
D_NOTFOUND_IO_START	INTEGER	
D_NOTFOUND_SYN_START	INTEGER	
S_FETCH_RET_START	INTEGER	
S_FETCH_UPD_START	INTEGER	
S_LB_ALLOK_START	INTEGER	
S_LB_GBNEEDLOCK_START	INTEGER	
S_LB_NEEDLOCK_START	INTEGER	
S_LB_OLDVER_START	INTEGER	
S_GB_NEEDLOCK_START	INTEGER	

S_GB_OLDVER_START	INTEGER	
S_NOTFOUND_IO_START	INTEGER	
S_NOTFOUND_SYN_START	INTEGER	
D_ASYNC_FETCH_START	INTEGER	
S_ASYNC_FETCH_START	INTEGER	
D_ASYNC_READIO_START	INTEGER	
S_ASYNC_READIO_START	INTEGER	
AS_READ_STALL_START	INTEGER	
AS_BATCH_WRITE_START	INTEGER	
AS_WRITE_STALL_START	INTEGER	
AREA_ITEMS_START	VARCHAR(128)	
AREA_ITEMS_START_STR_ID	INTEGER	STR_ID_DOMAIN
BIO_START	INTEGER	
DIO_START	INTEGER	
PAGEFAULTS_START	INTEGER	
PAGEFAULT_IO_START	INTEGER	
CPU_START	INTEGER	
CURRENT_PRIO_START	SMALLINT	
VIRTUAL_SIZE_START	INTEGER	
WS_SIZE_START	INTEGER	
WS_PRIVATE_START	INTEGER	
WS_GLOBAL_START	INTEGER	
CROSS_FAC_2_START	INTEGER	
CROSS_FAC_3_START	INTEGER	
CROSS_FAC_7_START	INTEGER	
CROSS_FAC_14_START	INTEGER	
DBS_READS_END	INTEGER	
DBS_WRITES_END	INTEGER	
RUJ_READS_END	INTEGER	
RUJ_WRITES_END	INTEGER	
AIJ_WRITES_END	INTEGER	
ROOT_READS_END	INTEGER	
ROOT_WRITES_END	INTEGER	
BUFFER_READS_END	INTEGER	
GET_VM_BYTES_END	INTEGER	
FREE_VM_BYTES_END	INTEGER	
LOCK_REQS_END	INTEGER	
REQ_NOT_QUEUED_END	INTEGER	
REQ_STALLS_END	INTEGER	
REQ_DEADLOCKS_END	INTEGER	
PROM_DEADLOCKS_END	INTEGER	
LOCK_RELS_END	INTEGER	

LOCK_STALL_TIME_END	INTEGER	
D_FETCH_RET_END	INTEGER	
D_FETCH_UPD_END	INTEGER	
D_LB_ALLOK_END	INTEGER	
D_LB_GBNEEDLOCK_END	INTEGER	
D_LB_NEEDLOCK_END	INTEGER	
D_LB_OLDVER_END	INTEGER	
D_GB_NEEDLOCK_END	INTEGER	
D_GB_OLDVER_END	INTEGER	
D_NOTFOUND_IO_END	INTEGER	
D_NOTFOUND_SYN_END	INTEGER	
S_FETCH_RET_END	INTEGER	
S_FETCH_UPD_END	INTEGER	
S_LB_ALLOK_END	INTEGER	
S_LB_GBNEEDLOCK_END	INTEGER	
S_LB_NEEDLOCK_END	INTEGER	
S_LB_OLDVER_END	INTEGER	
S_GB_NEEDLOCK_END	INTEGER	
S_GB_OLDVER_END	INTEGER	
S_NOTFOUND_IO_END	INTEGER	
S_NOTFOUND_SYN_END	INTEGER	
D_ASYNC_FETCH_END	INTEGER	
S_ASYNC_FETCH_END	INTEGER	
D_ASYNC_READIO_END	INTEGER	
S_ASYNC_READIO_END	INTEGER	
AS_READ_STALL_END	INTEGER	
AS_BATCH_WRITE_END	INTEGER	
AS_WRITE_STALL_END	INTEGER	
AREA_ITEMS_END	VARCHAR(128)	
AREA_ITEMS_END_STR_ID	INTEGER	STR_ID_DOMAIN
BIO_END	INTEGER	
DIO_END	INTEGER	
PAGEFAULTS_END	INTEGER	
PAGEFAULT_IO_END	INTEGER	
CPU_END	INTEGER	
CURRENT_PRIO_END	SMALLINT	
VIRTUAL_SIZE_END	INTEGER	
WS_SIZE_END	INTEGER	
WS_PRIVATE_END	INTEGER	
WS_GLOBAL_END	INTEGER	
CROSS_FAC_2_END	INTEGER	
CROSS_FAC_3_END	INTEGER	

CROSS_FAC_7_END	INTEGER
CROSS_FAC_14_END	INTEGER

Table 7−8 shows the REQUEST_BLR table.

Table 7−8 Columns for Table EPC$1_221_REQUEST_BLR

Column Name	Data Type	Domain
COLLECTION_RECORD_ID	SMALLINT	COLLECTION_RECORD_ID_DOMAIN
IMAGE_RECORD_ID	INTEGER	IMAGE_RECORD_ID_DOMAIN
CONTEXT_NUMBER	INTEGER	CONTEXT_NUMBER_DOMAIN
TIMESTAMP_POINT	DATE VMS	
CLIENT_PC	INTEGER	
STREAM_ID	INTEGER	
REQ_ID	INTEGER	
TRANS_ID	VARCHAR(16)	
TRANS_ID_STR_ID	INTEGER	STR_ID_DOMAIN
REQUEST_NAME	VARCHAR(31)	
REQUEST_NAME_STR_ID	INTEGER	STR_ID_DOMAIN
REQUEST_TYPE	INTEGER	
BLR	VARCHAR(127)	
BLR_STR_ID	INTEGER	STR_ID_DOMAIN

7.9.10 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
```

7.9.11 Using Oracle TRACE Collected Data

The following example shows how the OPTIMIZE AS clause is reflected in the Oracle TRACE database. When a trace collection is started the following SQL commands will record the request names.

```
SQL> attach `file personnel';
SQL> select last_name, first_name
cont> from employees
```

```
cont> optimize as request_one;
.
.
.
SQL> select employee_id
cont> from employees
cont> optimize as request_two;
.
.
.
SQL> select employee_id, city, state
cont> from employees
cont> optimize as request_three;
.
.
.
SQL> select last_name, first_name, employee_id, city, state
cont> from employees
cont> optimize as request_four;
.
.
.
```

Once an Oracle TRACE database has been populated from the collection, a query such as the following can be used to display the request names and types. The type values are described in Table 3−10. The unnamed queries in this example correspond to the queries executed by interactive SQL to validate the names of the tables an columns referenced in the user supplied queries.

```
SQL> select REQUEST_NAME, REQUEST_TYPE, TIMESTAMP_POINT
cont> from EPC$1_221_REQUEST_BLR;
REQUEST_NAME                      REQUEST_TYPE   TIMESTAMP_POINT
                                             1   15-JAN-1997 13:23:27.18
                                             1   15-JAN-1997 13:23:27.77
REQUEST_ONE                                  1   15-JAN-1997 13:23:28.21
REQUEST_TWO                                  1   15-JAN-1997 13:23:56.55
REQUEST_THREE                                1   15-JAN-1997 13:24:57.27
REQUEST_FOUR                                 1   15-JAN-1997 13:25:25.44
6 rows selected
```

The next example shows the internal query format (BLR) converted to SQL strings after EPC$EXAMPLES:EPC_BLR_TOSQL_CONVERTER.COM has been run.

```
SQL> SELECT A.REQUEST_NAME, B.SQL_STRING FROM
cont> EPC$1_221_REQUEST_BLR A,
cont> EPC$SQL_QUERIES B
cont> WHERE A.CLIENT_PC = 0 AND A.SQL_ID = B.SQL_ID;
A.REQUEST_NAME
  B.SQL_STRING
REQUEST_ONE
      SELECT C1.LAST_NAME, C1.FIRST_NAME.        FROM EMPLOYEES C1
. . .
REQUEST_TWO
      SELECT C1.EMPLOYEE_ID.        FROM EMPLOYEES C1
. . .
REQUEST_THREE
SELECT C1.EMPLOYEE_ID, C1.CITY, C1.STATE.        FROM EMPLOYEES C1
  .
  .
  .
4 rows selected
```

Table 4−17 shows the Request Types.

Table 7−9 Request Types

Symbolic Name	Value	Comment
RDB_K_REQTYPE_OTHER	0	A query executed internally by Oracle Rdb
RDB_K_REQTYPE_USER_REQUEST	1	A non−stored SQL statement, which includes compound statements
RDB_K_REQTYPE_PROCEDURE	2	A stored procedure
RDB_K_REQTYPE_FUNCTION	3	A stored function
RDB_K_REQTYPE_TRIGGER	4	A trigger action
RDB_K_REQTYPE_CONSTRAINT	5	A table or column constraint

7.9.12 AIP Length Problems in Indexes that Allow Duplicates

When an index allows duplicates, the length stored in the AIP will be 215 bytes, regardless of the actual index node size. Because an index with duplicates can have variable node sizes, the 215−byte size is used as a median length to represent the length of rows in the index's logical area.

When the row size in the AIP is less than the actual row length, it is highly likely that SPAM entries will show space is available on pages when they have insufficient space to store another full size row. This is the most common cause of insert performance problems.

For example, consider a case where an index node size of 430 bytes (a common default value) is used; the page size for the storage area where the index is stored is 2 blocks. After deducting page overhead, the available space on a 2−block page is 982 bytes. Assume that the page in this example is initially empty.

1. A full size (430−byte) index node is stored. As 8 bytes of overhead are associated with each row stored on a page, that leaves 982−430−8 = 544 free bytes remaining on the page.
2. A duplicate key entry is made in that index node and thus a duplicate node is created on the same page. An initial duplicate node is 112 bytes long (duplicate nodes can have a variety of sizes depending on when they are created, but for this particular example, 112 bytes is used). Therefore, 544−112−8 = 424 free bytes remain on the page.

At this point, 424 bytes are left on the page. That is greater than the 215 bytes that the AIP shows as the row length for the logical area, so the SPAM page shows that the page has space available. However, an attempt to store a full size index node on the page will fail, because the remaining free space (424 bytes) is not enough to store a 430−byte node.

In this case, another candidate page must be selected via the SPAM page, and the process repeats until a page that truly has sufficient free space available is found. In a logical area that contains many duplicate nodes, a significant percentage of the pages in the logical area may fit the scenario just described. When that is the case, and a new full size index node needs to be stored, many pages may need to be read and checked before one is found that can be used to store the row.

It is possible to avoid the preceding scenario by using logical area thresholds. The goal is to set a threshold such that the SPAM page will show a page is full when space is insufficient to store a full size index node.

Using the previous example, here is how to properly set logical area thresholds to prevent excessive pages checked on an index with a 430−byte node size that is stored on a 2−block page. To calculate the proper threshold value to use, you must first determine how full the page can get before no more full size nodes will fit on the page. In this example, a database page can have up to 982−430−8 = 544 bytes in use before the page is too full. Therefore, if 544 or fewer bytes are in use, then enough space remains to store another full size node. The threshold is then 544 / 982 = .553971, or 55%.

In addition, you can determine how full a page must be before a duplicate node of size 112 will no longer fit. In this example, a database page can have up to 982−112−8 = 862 bytes in use before the page is too full. Therefore, if 862 or fewer bytes are in use, then enough space remains to store another small duplicates node. The threshold is then 862 / 982 = .8778, or 88%.

Here is an example of creating an index with the above characteristics:

```
SQL> CREATE INDEX TEST_INDEX ON EMPLOYEES (LAST_NAME)
cont>     STORE IN RDB$SYSTEM
cont>     (THRESHOLD IS (55, 55, 88));
```

These settings mean that any page at over 55% full will not be fetched when inserting a full index node, however, it may be fetched when inserting the smaller duplicates node. When the page is over 88% full then neither a full node nor a duplicate node can be stored, so the page is set as FULL. The lowest setting is not used and so can be set to any value less than or equal to the lowest used threshold.

Note that the compression algorithm used on regular tables that have compression enabled does not apply to index nodes. Index nodes are not compressed like data rows and will always utilize the number of bytes that is specified in the node size. Do not attempt to take into account a compression factor when calculating thresholds for indexes.

7.9.13 RDM$BIND_MAX_DBR_COUNT Documentation Clarification

Appendix A in Oracle Rdb7 Guide to Database Performance and Tuning incorrectly describes the use of the RDM$BIND_MAX_DBR_COUNT logical name.

Following is an updated description. Note that the difference in actual behavior between what is in the existing documentation and the 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.

7.10 Oracle Rdb7 Guide to SQL Programming

This section provides information that is missing or changed in the Oracle Rdb7 Guide to SQL Programming.

7.10.1 Location of Host Source File Generated by the SQL Precompiler

When the SQL precompiler generates host source files (for example, .c, .pas, or .for) from the precompiler source files, it locates these files based on the Object qualifier in the command given to the SQL precompiler.

The following examples show the location where the host source file is generated.

When the Object qualifier is not specified on the command line, the object and the host source file take the name of the SQL precompiler with the extensions of .obj and .c, respectively. For example:

```
$ sqlpre/cc scc_try_mli_successful.sc
$ 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 the Object qualifier 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 a language other than C, it uses the appropriate host source extension (for example, .pas or .for). The files also default to the current directory if a directory specification is not specified. For example:

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

Directory MYDISK:[LUND]

SCC_TRY_MLI_SUCCESSFUL.SC;2

Total of 1 file.
$ dir myobj.*

Directory MYDISK:[LUND]

MYOBJ.C;1           MYOBJ.OBJ;2

Total of 2 files.

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

Directory MYDISK:[LUND]

SCC_TRY_MLI_SUCCESSFUL.SC;2

Total of 1 file.
```

```
$ 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.
```

7.10.2 Remote User Authentication

In the Oracle Rdb7 Guide to SQL Programming, Table 15−1 indicates that implicit authorization works from an OpenVMS platform to another OpenVMS platform using TCP/IP. This table is incorrect. Implicit authorization only works using DECnet in this situation.

The Oracle Rdb7 Guide to SQL Programming will be fixed in a future release.

7.10.3 Additional Information About Detached Processes

Oracle Rdb documentation omits necessary detail on running Oracle Rdb from a detached process.

Applications run from detached processes 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:

- Solution 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.
- Solution 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.

7.11 Guide to Using Oracle SQL/Services Client APIs

The following information describes Oracle SQL/Services documentation errors or omissions.

- The Guide to Using Oracle SQL/Services Client APIs does not describe changes to size and format of integer and floating−point data types
  Beginning with Oracle SQL/Services V5.1, the size and format of some integer and floating−point data types is changed as follows:
  - ♦ Trailing zeros occur in fixed−point numeric data types with SCALE FACTOR.
    Trailing zeros are now included after the decimal point up to the number of digits specified by the SCALE FACTOR. In versions of Oracle SQL/Services previous to V5.1, at most one trailing zero was included where the value was a whole number.
    The following examples illustrate the changes using a field defined as INTEGER(3):

```
V5.1 and     Versions previous
higher       to V5.1
--------     -----------------
  1.000        1.0
 23.400       23.4
567.890      567.89
```

  - ♦ Trailing zeros occur in floating−point data types. Trailing zeros are now included in the fraction, and leading zeros are included in the exponent, up to the maximum precision available, for fields assigned the REAL and DOUBLE PRECISION data types.

```
                                          Versions previous
Data Type           V5.1 and higher       to V5.1
----------------    ----------------------  -----------------
REAL                1.2340000E+01           1.234E+1
DOUBLE PRECISION    5.6789000000000000E+001  5.6789E+1
```

  - ♦ Size of TINYINT and REAL data types is changed.
    The maximum size of the TINYINT and REAL data types is changed to correctly reflect the precision of the respective data types.
    The following table shows the maximum lengths of the data types now and in previous versions:

```
                    V5.1 and    Versions previous
Data type           higher      to V5.1
----------          --------    -----------------
TINYINT              4           6
REAL                15          24
```

- The Guide to Using Oracle SQL/Services Client APIs does not describe that the sqlsrv_associate() service returns SQL error code −1028 when connecting to a database service if the user has not been granted the right to attach to the database.
  When a user connects to a database service, the sqlsrv_associate() service completes with the SQL error code −1028, SQL_NO_PRIV, if the user has been granted access to the Oracle SQL/Services service, but has not been granted the right to attach to the database. A record of the failure is written to the executor process's log file. Note that the sqlsrv_associate() service completes with the Oracle SQL/Services error code −2034, SQLSRV_GETACCINF if the user has not been granted access to the Oracle SQL/Services service.

Chapter 8
Known Problems and Restrictions

This chapter describes problems and restrictions relating to Oracle Rdb Release 7.1.5.2, and includes workarounds where appropriate.

8.1 Known Problems and Restrictions in All Interfaces

This section describes known problems and restrictions that affect all interfaces for Release 7.1.

8.1.1 Changes for Processing Existence Logical Names

A future release of Oracle Rdb will change the handling of so called "existence" logical names used to tune the Rdb environment. These existence logical names could in past versions be defined to any value to enable their effect. The Rdb documentation in most cases described using the value 1 or YES as that value and this change is upward compatible with the documentation.

In the future, Rdb will treat these logical names (see the list below) as Boolean logicals and accept a string starting with "Y", "y", "T", "t" or "1" to mean TRUE. All other values will be considered to be FALSE. This change will allow process level definitions to override definitions in higher logical name tables which was not possible previously.

Oracle recommends that customers examine all procedures that define the following logical names to ensure that their values conform to these rules prior to upgrading to Oracle Rdb V7.2.1.1 to avoid unexpected changes in behavior.

- RDMS$AUTO_READY
- RDMS$DISABLE_HIDDEN_KEY
- RDMS$DISABLE_MAX_SOLUTION
- RDMS$DISABLE_REVERSE_SCAN
- RDMS$DISABLE_TRANSITIVITY
- RDMS$DISABLE_ZIGZAG_BOOLEAN
- RDMS$ENABLE_BITMAPPED_SCAN
- RDMS$ENABLE_INDEX_COLUMN_GROUP
- RDMS$MAX_STABILITY
- RDMS$USE_OLD_COST_MODEL
- RDMS$USE_OLD_COUNT_RELATION
- RDMS$USE_OLD_SEGMENTED_STRING
- RDMS$USE_OLD_UPDATE_RULES

8.1.2 SQL Module or Program Fails with %SQL−F−IGNCASE_BAD

Bug 2351248

A SQL module or pre−compiled SQL program built with Rdb Release 6.1 or earlier may fail when running under Rdb Release 7.1 if the program submits queries that involve certain kinds of character operations on parameters in the queries. For example, a LIKE operator in the WHERE clause of a SQL statement requires SQL to look for character− or string−matching wildcard characters. Another example is the use of IGNORE CASE, which causes SQL to equivalence upper and lower case characters for the character set in use.

The following example shows a portion of a SQL module language program that queries a PERSONNEL database.

```
DECLARE MANL_NAME_LIST CURSOR FOR
 SELECT DISTINCT E.LAST_NAME,E.FIRST_NAME,J.JOB_CODE,J.DEPARTMENT_CODE,E.CITY
FROM   DB1_HANDLE.EMPLOYEES E,DB1_HANDLE.JOB_HISTORY J
 WHERE J.EMPLOYEE_ID = E.EMPLOYEE_ID
   AND E.STATUS_CODE = STATUS_CODE
   AND E.CITY LIKE CITYKEY IGNORE CASE
   ORDER BY E.EMPLOYEE_ID DESC, E.LAST_NAME DESC

PROCEDURE SQL_OPN_NAME_LIST
SQLCODE
CITYKEY       CHAR(20)
STATUS_CODE   CHAR(1);
OPEN MANL_NAME_LIST;
```

If the SQL module containing the code above is compiled and linked into an executable using a pre−7.0 version of Rdb, it will run properly against that version. However, if the same program is run in an Rdb 7.1 environment, a call to the SQL_OPN_NAME_LIST procedure will return a SQLCODE of −1. The RDB$MESSAGE_VECTOR will contain a code associated with the following message:

```
%SQL−F−IGNCASE_BAD, IGNORE CASE not supported for character set
```

As a workaround to this problem, re−link the program using a 7.1 version of SQL$INT.EXE and/or SQL$USER.OLB.

8.1.3 Domain−qualified TCP/IP Node Names in Distributed Transactions

Bug 3735144

When using TCP/IP for Oracle Rdb remote connections, distributed transactions involving databases on nodes which are not on the same subnet may not work.

Remote Rdb has the capability to make remote connections via TCP/IP in lieu of DECnet. (See the Oracle Rdb OpenVMS Installation and Configuration Guide for how to set this up.) However, distributed transactions involving remote databases connected via TCP/IP have been difficult. This is because Rdb relies on OpenVMS DECdtm for distributed transaction support and DECdtm requires DECnet for off−node communication. (This is an OpenVMS and not an Rdb restriction. Contact Hewlett−Packard OpenVMS Support for more details.)

OpenVMS provides a capability to run DECnet over TCP/IP so that OpenVMS services which require DECnet (like DECdtm) can operate in an environment where a TCP/IP network is used as the communications backbone. This capability allows DECdtm (and hence Rdb) to manage distributed transactions via TCP/IP. (See HP's OpenVMS DECnet−Plus documentation set for how to configure and use this capability.)

However, for a transaction involving a remote database, Rdb only provides the SCSNODE name of the remote node to DECdtm. For example, consider the following SQL attaches to two remote databases using TCP/IP:

```
SQL>attach 'alias db1 filename node1.a.b.c::db_root:db1 user ''me'' using
''pw''';
SQL>attach 'alias db2 filename node1.a.b.c::db_root:db2 user ''me'' using
''pw''';
```

In the above example, Rdb can successfully connect to both remote databases using the TCP/IP address "node1.a.b.c." but when multiple databases are attached, Rdb implicitly uses distributed transactions via DECdtm. Since Rdb only passes DECdtm the SCSNODE name retrieved from the RDBSERVERnn at the other end of the connection, DECdtm does not, in general, have the information it needs to resolve the remote reference. It will only be able to do so if the SCSNODE name and the TCP/IP node name are the same and the local node is on the same subnet (i.e. ".a.b.c" in the example). Otherwise, after the second attach is made, the following error message will be received as soon as a transaction is started:

```
SQL>set trans read write;
%RDB-F-SYS_REQUEST_CAL, error from system services request - called from 100001
-RDB-E-DECDTMERR, DECdtm system service call error
-IPC-E-BCKTRNSFAIL, failure on the back translate address request
```

WORKAROUND

There are three potential workarounds:

1. If distributed transactions are unimportant to the application, they can be disabled by defining the logical name SQL$DISABLE_CONTEXT to TRUE. Rdb will then not call DECdtm and the node name resolution problem will not be seen. However, it will be the problem of the application to maintain database integrity in the event that a commit succeeds on one database and not on another. See the Rdb Guide to Distributed Transactions for more information.
2. If all the nodes involved in the distributed transaction are in the same domain, then TCP/IP can resolve the node with only the first part of the node provided that the SCSNODE name is identical to it. In the example above, this would mean that the remote node had an SCSNODE name of "NODE1" and that the local node was on TCP/IP subnet ".a.b.c".
3. It may also be possible to define a DNS/BIND alias name for the remote node's SCSNODE name to the local node's TCP/IP database. This should allow the SCSNODE name passed by Rdb Dispatch to be translated successfully. For example, assuming HP TCP/IP Services for OpenVMS is the TCP/IP protocol stack then a command like the following could be used on the local node:

   ```
   $ TCP SET HOST NODE1.A.B.C/address=nnn.nnn.nnn.nnn/alias=NODE1_SCS
   ```

   Where "nnn.nnn.nnn.nnn" is the IP address and "NODE1_SCS" the OpenVMS SCSNODE name of the remote node. See the HP DECnet-Plus documentation set for more information on how to maintain TCP/IP domain databases.

8.1.4 Some SQL Dialect-required Warnings not Delivered

Bugs 3651847 and 4532451

The required warnings (information codes) for such things as rows eliminated for nulls (%RDB-I-ELIM_NULL) and string truncation (%RDB-I-TRUN_RTRV) are not being returned for singleton SELECT and singleton UPDATE statements (statements that return a single row using the INTO clause). To demonstrate with a PERSONNEL database, use the following interactive SQL commands:

```
SQL> set dialect 'sql92';
SQL> attach 'filename sql$database';
SQL>
SQL> ! Force a row to contain NULL for SALARY_AMOUNT
SQL> update salary_history
cont> set salary_amount = NULL
cont> where employee_id = '00471'
cont>   and salary_end = date vms'20-Aug-1981';
1 row updated
SQL>
SQL> declare :avg_sal integer(2);
SQL>
SQL> ! No informational generated (but is expected)
SQL> select avg(salary_amount) into :avg_sal
cont> from salary_history where employee_id = '00471'
cont>  and salary_end >= date vms'1-AUG-1970';
SQL> show sqlca
SQLCA:
        SQLCAID:        SQLCA           SQLCABC:        128
        SQLCODE:        0
        SQLERRD:        [0]: 0
                        [1]: 0
                        [2]: 1
                        [3]: 0
                        [4]: 0
                        [5]: 0
        SQLWARN0:       0       SQLWARN1:       0       SQLWARN2:       0
        SQLWARN3:       0       SQLWARN4:       0       SQLWARN5:       0
        SQLWARN6:       0       SQLWARN7:       0
        SQLSTATE:       00000
SQL> print :avg_sal;
        AVG_SAL
       60893.86
SQL>
SQL> ! Non singleton query returns correct informational
SQL> select avg(salary_amount)
cont> from salary_history where employee_id = '00471'
cont>  and salary_end >= date vms'1-AUG-1970';

  6.089385714285714E+004
1 row selected
%RDB-I-ELIM_NULL, null value eliminated in set function
SQL> show sqlca
SQLCA:
        SQLCAID:        SQLCA           SQLCABC:        128
        SQLCODE:        1003
        SQLERRD:        [0]: 0
                        [1]: 0
                        [2]: 1
                        [3]: 0
                        [4]: 0
                        [5]: 0
        SQLWARN0:       0       SQLWARN1:       0       SQLWARN2:       0
        SQLWARN3:       0       SQLWARN4:       0       SQLWARN5:       0
        SQLWARN6:       0       SQLWARN7:       0
        SQLSTATE:       01003
%RDB-I-ELIM_NULL, null value eliminated in set function
SQL>
SQL> rollback;
```

Since there is a row in the SALARY_HISTORY table with a NULL in SALARY_AMOUNT, the set function

AVG should report an informational message (and return a special warning level SQLSTATE/SQLCODE value).

```
%RDB-I-ELIM_NULL, null value eliminated in set function
```

8.1.5 Partitioned Index with Descending Column and Collating Sequence

Bug 2797443

A known problem exists in which a query can return wrong results (number of rows returned is incorrect). This can happen on a table that has a multi−column, partitioned index in which one of the columns is sorted in descending order and the column has an associated collating sequence.

The following example can be used to demonstrate the problem.

```
$ sql$

create database file mf_collating.rdb alloc 10
  collating sequence french french
  create storage area area1 alloc 10
  create storage area area2 alloc 10
  create storage area area3 alloc 10;

create table tab1 (id tinyint, r3 char (3));

insert into tab1 (id, r3) values (1, 'a');
1 row inserted
insert into tab1 (id, r3) values (1, 'b');
1 row inserted
insert into tab1 (id, r3) values (1, 'f');
1 row inserted

create index y3 on tab1 (id asc, r3 desc)
       store using (id, r3)
       in area1 with limit of (1, 'k')
       in area2 with limit of (1, 'e')
       otherwise in area3 ;

commit;

set flags 'strategy';

! Here is a query that returns the correct rows using sequential rather
! than indexed access.

select id, r3 from tab1 where id = 1 and r3 <= 'e'
  optimize for sequential access;
Conjunct        Get     Retrieval sequentially of relation TAB1
   ID   R3
    1   a
    1   b
2 rows selected

! Here is the same query without the sequential access restriction.
! Note in the query strategy that index Y3 is used for data retrieval.
```

```
! This query ought to (but does not) return the same set of rows as
! for the sequential access query.

select id, r3 from tab1 where id = 1 and r3 <= 'e';
Leaf#01 FFirst TAB1 Card=3
  BgrNdx1 Y3 [2:1] Fan=16
0 rows selected
```

8.1.6 RDO IMPORT Does Not Support FORWARD_REFERENCES Created by SQL EXPORT

Recent versions of SQL EXPORT have included support for new features as they are added to Oracle Rdb. In particular, SQL now generates forward references for routines to allow references in the metadata prior to those routines being created. No such enhancements will be made to RDO IMPORT.

Due to changes in the CREATE STORAGE MAP statement for this release, the RDO IMPORT command is no longer able to process interchange (.RBR) files created by SQL EXPORT due to forward references to new storage mapping routines. See the Oracle Rdb 7.1.4 Release Notes, Enhancement Chapter for details.

Note

The RDO IMPORT command has been deprecated since the release of Oracle Rdb V7.0. Oracle recommends that users change all scripts to use the SQL IMPORT command in the future.

The following example shows the reported error:

```
$ RDO
IMPORT 'thresh_alter_sql' INTO 'thresh' DICTIONARY IS NOT USED.
%RDO-W-UNSIMPORT, RDO IMPORT does not support all Oracle Rdb features, please
use SQL IMPORT
Exported by Oracle Rdb V7.1-301 Import/Export utility
A component of Oracle Rdb SQL V7.1-301
   .
   .
   .
IMPORTing STORAGE AREA: RDB$SYSTEM
IMPORTing STORAGE AREA: AREA1
IMPORTing STORAGE AREA: AREA2
IMPORTing STORAGE AREA: DEFAULT_AREA
%RDO-E-EXTRADATA, unexpected data at the end of the RBR file
```

With the current release, you can work around this problem in one of the following ways:

1. Change the command to execute the SQL IMPORT command. This is the recommended and long term solution to this problem.
2. Change the SQL EXPORT command to include the NO FORWARD_REFERENCES clause. This will eliminate the definitions which currently cause errors in RDO IMPORT. However, this interchange file may then not contain sufficient information to fully import the database.
3. The RMU/LOAD command can also be used to extract the data for individual tables. You must use the /MATCH_NAME qualifier for Load.

8.1.7 New Attributes Saved by RMU/LOAD Incompatible With Prior Versions

Bug 2676851

To improve the behavior of unloading views, Oracle Rdb Release 7.1.2 changed the way view columns were unloaded so that attributes for view computed columns, COMPUTED BY and AUTOMATIC columns were saved. These new attributes are not accepted by prior releases of Oracle Rdb.

The following example shows the reported error trying to load a file from V7.1.2 under V7.1.0.4.

```
%RMU-F-NOTUNLFIL, Input file was not created by RMU UNLOAD
%RMU-I-DATRECSTO,   0 data records stored.
%RMU-F-FTL_LOAD, Fatal error for LOAD operation at 21-OCT-2003 16:34:54.20
```

You can workaround this problem by using the /RECORD_DEFINITION qualifier and specifying the FORMAT=DELIMITED option. However, this technique does not support LIST OF BYTE VARYING column unloading.

8.1.8 RDMS-E-RTNSBC_INITERR, Cannot init. external routine server site executor

Execution of an external function or procudure with server site binding may unexpectedly fail.

The following example shows this problem.

```
%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
```

In this example, NNNNNNNNN is the function name and XX is a decimal value such as 41.

While such errors are possible they are very unlikely to be seen, especially on systems that have had 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, as in the following example.

```
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
```

8.1.9 SYSTEM−F−INSFMEM Fatal Error With SHARED MEMORY IS SYSTEM or LARGE MEMORY IS ENABLED in Galaxy Environment

When using the GALAXY SUPPORT IS ENABLED feature in an OpenVMS Galaxy environment, a *%SYSTEM−F−INSFMEM, insufficient dynamic memory error* may be returned when mapping record caches or opening the database. One source of this problem specific to a Galaxy configuration is running out of Galaxy Shared Memory regions. For Galaxy systems, GLX_SHM_REG is the number of shared memory region structures configured into the Galaxy Management Database (GMDB).

While the default value (for OpenVMS versions through at least V7.3−1) of 64 regions might be adequate for some installations, sites using a larger number of databases or row caches when the SHARED MEMORY IS SYSTEM or LARGE MEMORY IS ENABLED features are enabled may find the default insufficient.

If a *%SYSTEM−F−INSFMEM, insufficient dynamic memory* error is returned when mapping record caches or opening databases, Oracle Corporation recommends that you increase the GLX_SHM_REG parameter by 2 times the sum of the number of row caches and number of databases that might be accessed in the Galaxy at one time. As the Galaxy shared memory region structures are not very large, setting this parameter to a higher than required value does not consume a significant amount of physical memory. It also may avoid a later reboot of the Galaxy environment. This parameter must be set on all nodes in the Galaxy.

Galaxy Reboot Required

Changing the GLX_SHM_REG system parameter requires that the OpenVMS Galaxy environment be booted from scratch. That is, all nodes in the Galaxy must be shut down and then the Galaxy reformed by starting each instance.

8.1.10 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.

8.1.11 Optimization of Check Constraints

Bug 1448422

When phrasing constraints using the "CHECK" syntax, a poorer strategy can be chosen by the optimizer than when the same or similar constraint is phrased using referential integrity (PRIMARY and FOREIGN KEY) constraints.

For example, I have two tables T1 and T2, both with one column, and I wish to ensure that all values in table T1 exist in T2. Both tables have an index on the referenced field. I could use a PRIMARY KEY constraint on T2 and a FOREIGN KEY constraint on T1.

```
SQL> alter table t2
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, 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 like this:

```
SQL> delete from t2 where f2=1;
Get     Temporary relation      Retrieval by index of relation T2
  Index name  I2 [1:1]
Cross block of 2 entries
  Cross block entry 1
    Index only retrieval of relation T1
      Index name  I1 [0:0]
  Cross block entry 2
    Conjunct        Aggregate-F1    Conjunct
    Index only retrieval of relation T2
      Index name  I2 [1:1]
%RDB-E-INTEG_FAIL, violation of constraint T1_CHECK1 caused operation to fail
```

While the entire T1 index is scanned, at least the value from T1 is used to perform an index lookup on T2.

These restrictions result from semantic differences in the behavior of the "IN" and "EXISTS" operators with respect to null handling, and the complexity of dealing with non−equality join conditions.

To improve the performance of this type of integrity check on larger tables, it is possible to use a series of triggers to perform the constraint check. The following triggers perform a similar check to the constraints above.

```
SQL> create trigger t1_insert
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 NULL to indicate the absence of a value, the above triggers could easily be modified to allow for these semantics.

8.1.12 Using Databases from Releases Earlier Than V6.0

You cannot convert or restore databases earlier than V6.0 directly to V7.1. The RMU Convert command for V7.1 supports conversions from V6.0 through V7.0 only. If you have a V3.0 through V5.1 database, you must convert it to at least V6.0 and then convert it to V7.1. For example, if you have a V4.2 database, convert it first to at least V6.0, then convert the resulting database to V7.1.

If you attempt to convert a database created prior to V6.0 directly to V7.1, Oracle RMU generates an error.

8.1.13 Carryover Locks and NOWAIT Transaction Clarification

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

8.1.14 Unexpected Results Occur During Read−Only Transactions on a Hot Standby Database

When using Hot Standby, it is typical to use the standby database for reporting, simple queries, and other read−only transactions. If you are performing these types of read−only transactions on a standby database, be sure you can tolerate a READ COMMIT level of isolation. This is because the Hot Standby database might be updated by another transaction before the read−only transaction finishes, and the data retrieved might not be what you expected.

Because Hot Standby does not write to the snapshot files, the isolation level achieved on the standby database for any read−only transaction is a READ COMMITED transaction. This means that nonrepeatable reads and phantom reads are allowed during the read−only transaction:

- Nonrepeatable read operations: Allows the return of different results within a single transaction when an SQL operation reads the same row in a table twice. Nonrepeatable reads can occur when another transaction modifies and commits a change to the row between transactions. Because the standby database will update the data when it confirms a transaction has been committed, it is very possible to see an SQL operation on a standby database return different results.
- Phantom read operations: Allows the return of different results within a single transaction when an SQL operation retrieves a range of data values (or similar data existence check) twice. Phantoms can occur if another transaction inserted a new record and committed the insertion between executions of the range retrieval. Again, because the standby database may do this, phantom reads are possible.

Thus, you cannot rely on any data read from the standby database to remain unchanged. Be sure your read−only transactions can tolerate a READ COMMIT level of isolation before you implement procedures that read and use data from a standby database.

8.1.15 Both Application and Oracle Rdb Using SYS$HIBER

In application processes that use Oracle Rdb and the $HIBER system service (possibly through RTL routines such as LIB$WAIT), the application must ensure that the event being waited for has actually occurred. Oracle Rdb uses $HIBER/$WAKE sequences for interprocess communications particularly when the ALS (AIJ Log Server) feature is enabled.

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

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

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

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

The LIB$K_NOWAKE flag can be specified when using the OpenVMS LIB$WAIT routine to allow an alternate wait scheme (using the $SYNCH system service) that can avoid potential problems with multiple code sequences using the $HIBER system service.

8.1.16 Bugcheck Dump Files with Exceptions at COSI_CHF_SIGNAL

In certain situations, Oracle Rdb bugcheck dump files 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 is needed when working with Oracle Rdb Worldwide Support.

8.1.17 Read−only Transactions Fetch AIP Pages Too Often

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

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

This problem has been reduced in Oracle Rdb release 7.0. The AIP entries are now read once and then are not read again unless they need to be. This optimization requires that the carry−over locks feature be enabled (this is the default setting). If carry over locks are not enabled, this optimization is not enabled and the behavior is the same as in previous releases.

8.1.18 Row Cache Not Allowed While Hot Standby Replication is Active

The row cache feature may not be enabled on a hot standby database while replication is active. The hot standby feature will not start if row cache is enabled.

This restriction exists because rows in the row cache are accessed via logical dbkeys. However, information transferred to the standby database via the after image journal facility only contains physical dbkeys. Because there is no way to maintain rows in the cache via the hot standby processing, the row cache must be disabled when the standby database is open and replication is active.

A new command qualifier, ROW_CACHE=DISABLED, has been added to the RMU Open command. To open the hot standby database prior to starting replication, use the ROW_CACHE=DISABLED qualifier on the RMU Open command.

8.1.19 Excessive Process Page Faults and other Performance Considerations During Oracle Rdb Sorts

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

Oracle Rdb includes its own copy of the OpenVMS SORT32 code within the Oracle Rdb images and does not generally call the routines in the OpenVMS run−time library. A copy of the SORT32 code is used to provide stability between versions of Oracle Rdb and OpenVMS and because Oracle Rdb calls the sort routines from executive processor mode which is difficult to do using the SORT32 shareable image. SQL IMPORT and RMU Load operations do, however, call the OpenVMS SORT run−time library.

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

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

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

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

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

Sort work files are another factor to consider when tuning for Oracle Rdb sort operations. When the operation can not be done in the available memory, SORT uses temporary disk files to hold the data as it is being sorted. The Oracle Rdb7 Guide to Database Performance and Tuning contains more detailed information about sort work files.

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

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

Note that SORT uses the work files for different sorted runs, and then merges the sorted runs into larger groups. If the source data is mostly sorted, then not every sort work file may need to be accessed. This is a possible source of confusion because even with 10 sort work files, it is possible to exceed the capacity of the first SORT file and the sort operation fails 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 allocation within Oracle Rdb.

8.1.20 Control of Sort Work Memory Allocation

Oracle Rdb uses a built−in SORT32 package to perform many sort operations. Sometimes, these sorts exhibit a significant performance problem when initializing work memory to be used for the sort. This behavior can be experienced, for example, when a very large sort cardinality is estimated, but the actual sort cardinality is small.

In rare cases, it may be desirable to artificially limit the sort package's use of work memory. Two logicals have been created to allow this control. In general, there should be no need to use either of these logicals and misuse of them can significantly impact sort performance. Oracle recommends that these logicals be used carefully and sparingly.

The logical names are:

Table 8−1 Sort Memory Logicals

Logical	Definition
RDMS$BIND_SORT_MEMORY_WS_FACTOR	Specifies a percentage of the process's working set limit to be used when allocating sort memory for the built−in SORT32 package. If not defined, the default value is 75 (representing 75%), the maximum value is 75 (representing 75%), and the minimum value is 2 (representing 2%). Processes with vary large working set limits can sometimes experience significant page faulting and CPU consumption while initializing sort memory. This logical name can restrict the sort work memory to a percentage of the processes maximum working set.
RDMS$BIND_SORT_MEMORY_MAX_BYTES	Specifies an absolute limit to be used when allocating sort memory for the built−in SORT32 package. If not defined, the default value is unlimited (up to 1GB), the maximum value is 2,147,483,647 and the minimum value is 32,768.

8.1.21 The Halloween Problem

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

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

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

CONCURRENCY/CONSISTENCY options in RDO/RDML.

Oracle Rdb avoids this problem if it knows that the cursors subject table will be updated. For example, if the SQL syntax UPDATE ... WHERE CURRENT OF is used to perform updates of target rows, or the RDO/RDML MODIFY statement uses the context variable for the stream. Then the optimizer will choose an alternate access strategy if an update can occur which may cause the Halloween problem. This can be seen in the access strategy in Example 2−2 as a "Temporary relation" being created to hold the result of the cursor query.

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

The following example shows that the EMP_LAST_NAME index is used for retrieval. Any update performed will possibly be subject to the Halloween problem.

```
SQL> set flags 'strategy';
SQL> declare emp  cursor for
cont> select * from employees where last_name >= 'M'
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;
```

The following example shows that the query specifies that the column LAST_NAME will be updated by some later query. Now the optimizer protects the EMP_LAST_NAME index used for retrieval by using a "Temporary Relation" to hold the query result set. Any update performed on LAST_NAME will now avoid the Halloween problem.

```
SQL> set flags 'strategy';
SQL> declare emp2 cursor for
cont> select * from employees where last_name >= 'M'
cont> order by last_name
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 cannot perform the normal protection against the Halloween problem.

8.2 SQL Known Problems and Restrictions

This section describes known problems and restrictions for the SQL interface for release 7.1.

8.2.1 Interchange File (RBR) Created by Oracle Rdb Release 7.1 Not Compatible With Previous Releases

To support the large number of new database attributes and objects, the protocol used by SQL EXPORT and SQL IMPORT has been enhanced to support more protocol types. Therefore, this format of the Oracle Rdb release 7.1 interchange files can no longer be read by older versions of Oracle Rdb.

Oracle Rdb continues to provide upward compatibility for interchange files generated by older versions.

Oracle Rdb has never supported backward compatibility, however, it was sometimes possible to use an interchange file with an older version of IMPORT. However, this protocol change will no longer permit this usage.

8.2.2 System Relation Change for International Database Users

Due to an error in creating the RDB$FIELD_VERSIONS system relation, another system relation, RDB$STORAGE_MAP_AREAS, cannot be accessed if the session character sets are not set to DEC_MCS.

This problem prevents the new Oracle Rdb GUIs, specifically the Oracle Rdb Schema Manager, from viewing indexes and storage maps from existing Oracle Rdb databases.

The problem can be easily corrected by executing the following SQL statement after attaching to the database:

```
SQL> UPDATE RDB$FIELD_VERSIONS SET RDB$FIELD_SUB_TYPE = 32767
cont> WHERE RDB$FIELD_NAME = 'RDB$AREA_NAME';
```

8.2.3 Single Statement LOCK TABLE is Not Supported for SQL Module Language and SQL Precompiler

The new LOCK TABLE statement is not currently supported as a single statement within the module language or embedded SQL language compiler.

Instead you must enclose the statement in a compound statement. That is, use BEGIN... END around the statement as shown in the following example. This format provides all the syntax and flexibility of LOCK TABLE.

This restriction does not apply to interactive or dynamic SQL.

The following extract from the module language listing file shows the reported error if you use LOCK TABLE as a single statement procedure. The other procedure in the same module is acceptable because it uses a compound statement that contains the LOCK TABLE statement.

```
1 MODULE sample_test
2 LANGUAGE C
3 PARAMETER COLONS
4
5 DECLARE ALIAS FILENAME 'mf_personnel'
6
7 PROCEDURE a (SQLCODE);
8 LOCK TABLE employees FOR EXCLUSIVE WRITE MODE;
%SQL-F-WISH_LIST, (1) Feature not yet implemented - LOCK TABLE requires compound
statement
9
10 PROCEDURE b (SQLCODE);
11 BEGIN
12 LOCK TABLE employees FOR EXCLUSIVE WRITE MODE;
13 END;
```

To workaround this problem of using LOCK TABLE for SQL module language or embedded SQL application, use a compound statement in an EXEC SQL statement.

8.2.4 Multistatement or Stored Procedures May Cause Hangs

Long−running multistatement or stored procedures can cause other users in the database to hang if the procedures obtain resources needed by those other users. Some resources obtained by the execution of a multistatement or stored procedure are not released until the multistatement or stored procedure finishes. Thus, any−long running multistatement or stored procedure can cause other processes to hang. This problem can be encountered even if the statement contains SQL COMMIT or ROLLBACK statements.

The following example demonstrates the problem. The first session enters an endless loop; the second session attempts to backup the database but hangs forever.

```
Session 1:

SQL> attach 'filename MF_PERSONNEL';
SQL> create function LIB$WAIT (in real by reference)
cont> returns integer;
cont> external name LIB$WAIT location 'SYS$SHARE:LIBRTL.EXE'
cont> language general general parameter style variant;
SQL> commit;
        .
        .
        .
$ SQL
SQL> attach 'filename MF_PERSONNEL';
SQL> begin
cont> declare :LAST_NAME LAST_NAME_DOM;
cont> declare :WAIT_STATUS integer;
cont> loop
cont> select LAST_NAME into :LAST_NAME
cont> from EMPLOYEES where EMPLOYEE_ID = '00164';
cont> rollback;
cont> set :WAIT_STATUS = LIBWAIT (5.0);
cont> set transaction read only;
cont> end loop;
cont> end;

Session 2:
```

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

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

$ RMU/SHOW LOCKS /MODE=BLOCKING MF_PERSONNEL
    .
    .
    .
Resource: nowait signal

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

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

8.2.5 Use of Oracle Rdb from Shareable Images

If code in the image initialization routine of a shareable image makes any calls into Oracle Rdb, through SQL or any other means, access violations or other unexpected behavior may occur if Oracle Rdb images have not had a chance to do their own initialization.

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

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

This is not a bug; it is a restriction resulting from the way OpenVMS image activation works.

8.3 Oracle RMU Known Problems and Restrictions

This section describes known problems and restrictions for the RMU interface for release 7.1.

8.3.1 RMU/BACKUP MAX_FILE_SIZE Option Has Been Disabled

The MAX_FILE_SIZE option of the RMU/BACKUP/DISK_FILE qualifier for backup to multiple disk files has been temporarily disabled since it creates corrupt RBF files if the maximum file size in megabytes is exceeded and a new RBF file is created. It also does not give a unique name to the new RBF file but creates an RBF file with the same name but a new version number in the same disk directory. This will cause an RMU−F−BACFILCOR error on the restore and the restore will not complete.

The multi−file disk backup and restore will succeed if this option is not used. If this option is specified, a warning message is now output that this qualifier will be ignored.

The following example shows that the MAX_FILE_SIZE option, when used with the /DISK_FILE qualifier on an RMU/BACKUP, will be ignored and a warning message will be output.

```
$ RMU/BACKUP /ONLINE                    −
             /NOCRC                     −
             /NOLOG                     −
             /NOINCREMENTAL             −
             /QUIET_POINT               −
             TEST_DB_DIR:TEST_DB
  −
BACKUP_DIR_1:TEST_DB/DISK_FILE=(WRITER_THREADS=3,MAX_FILE_SIZE=10) ,−
BACKUP_DIR_2:/DISK_FILE=(WRITER_THREADS=3,MAX_FILE_SIZE=10) ,−
BACKUP_DIR_3:/DISK_FILE=(WRITER_THREADS=3,MAX_FILE_SIZE=10)

%RMU−W−DISABLEDOPTION, The MAX_FILE_SIZE option is temporarily disabled
        and will be ignored
```

As a workaround to avoid this problem, do not specify the MAX_FILE_SIZE option with the /DISK_FILE qualifier.

8.3.2 RMU Convert Fails When Maximum Relation ID is Exceeded

If, when relation IDs are assigned to new system tables during an RMU Convert of an Oracle Rdb V7.0 database to a V7.1 database, the maximum relation ID of 8192 allowed by Oracle Rdb is exceeded, the fatal error %RMU−F−RELMAXIDBAD is displayed and the database is rolled back to V70. Contact your Oracle support representative if you get this error. Note that when the database is rolled back, the fatal error %RMU−F−CVTROLSUC is displayed to indicate that the rollback was successful but caused by the detection of a fatal error and not requested by the user.

This condition only occurs if there are an extremely large number of tables defined in the database or if a large number of tables were defined but have subsequently been deleted.

The following example shows both the %RMU−F−RELMAXIDBAD error message if the allowed database relation ID maximum of 8192 is exceeded and the %RMU−F−CVTROLSUC error message when the database has been rolled back to V7.0 since it cannot be converted to V7.1:

```
$rmu/convert mf_personnel
%RMU−I−RMUTXT_000, Executing RMU for Oracle Rdb V7.1−00
Are you satisfied with your backup of
 DEVICE:[DIRECTORY]MF_PERSONNEL.RDB;1 and your backup of
 any associated .aij files [N]? Y
 %RMU−I−LOGCONVRT, database root converted to current structure level
 %RMU−F−RELMAXIDBAD, ROLLING BACK CONVERSION − Relation ID exceeds maximum
  8192 for system table RDB$RELATIONS
 %RMU−F−CVTROLSUC, CONVERT rolled−back for
  DEVICE:[DIRECTORY]MF_PERSONNEL.RDB;1 to version V7.0
```

The following example shows the normal case when the maximum allowed relation ID is not exceeded:

```
$rmu/convert mf_personnel
%RMU−I−RMUTXT_000, Executing RMU for Oracle Rdb V7.1−00
Are you satisfied with your backup of
 DEVICE:[DIRECTORY]MF_PERSONNEL.RDB;1 and your backup of
 any associated .aij files [N]? Y
%RMU−I−LOGCONVRT, database root converted to current structure level
%RMU−S−CVTDBSUC, database DEVICE:[DIRECTORY]MF_PERSONNEL.RDB;1
 successfully converted from version V7.0 to V7.1
%RMU−I−CVTCOMSUC, CONVERT committed for
 DEVICE:[DIRECTORY]MF_PERSONNEL.RDB;1 to version V7.1
```

8.3.3 RMU Unload /After_Journal Requires Accurate AIP Logical Area Information

The RMU Unload /After_Journal command uses the on−disk area inventory pages (AIPs) to determine the appropriate type of each logical area when reconstructing logical dbkeys for records stored in mixed−format storage areas. However, the logical area type information in the AIP is generally unknown for logical areas created prior to Oracle Rdb release 7.0.1. If the RMU Unload /After_Journal command cannot determine the logical area type for one or more AIP entries, a warning message is displayed for each such area and may ultimately return logical dbkeys with a 0 (zero) area number for records stored in mixed−format storage areas.

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

EMPLOYEES /TYPE=TABLE

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

EMPLOYEES /AREA=EMPID_OVER /TYPE=TABLE

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

- TABLE
  Specifies that the logical area is a data table. This would be a table created using the SQL CREATE TABLE syntax.
- B−TREE
  Specifies that the logical area is a B−tree index. This would be an index created using the SQL CREATE INDEX TYPE IS SORTED syntax.
- HASH
  Specifies that the logical area is a hash index. This would be an index created using the SQL CREATE INDEX TYPE IS HASHED syntax.
- SYSTEM
  Specifies that the logical area is a system record that is used to identify hash buckets. Users cannot explicitly create these types of logical areas.

Note

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

- BLOB
  Specifies that the logical area is a BLOB repository.

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

8.3.4 Do Not Use HYPERSORT with RMU Optimize After_Journal Command

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

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

Because of this, the use of the high−performance Sort/Merge utility is not supported for the RMU Optimize After_Journal command. Do not define the logical name SORTSHR to reference HYPERSORT.EXE.

8.3.5 Changes in EXCLUDE and INCLUDE Qualifiers for RMU Backup

The RMU Backup command no longer accepts both the Include and Exclude qualifiers in the same command. This change removes the confusion over exactly what gets backed up when Include and Exclude are specified on the same line, but does not diminish the capabilities of the RMU Backup command.

To explicitly exclude some storage areas from a backup, use the Exclude qualifier to name the storage areas to be excluded. This causes all storage areas to be backed up except for those named by the Exclude qualifier.

Similarly, the Include qualifier causes only those storage areas named by the qualifier to be backed up. Any storage area not named by the Include qualifier is not backed up. The Noread_only and Noworm qualifiers continue to cause read−only storage areas and WORM storage areas to be omitted from the backup even if these areas are explicitly listed by the Include qualifier.

Another related change is in the behavior of EXCLUDE=*. In previous versions, EXCLUDE=* caused all storage areas to be backed up. Beginning with V7.1, EXCLUDE=* causes only a root backup to be done. A backup created by using EXCLUDE=* can be used only by the RMU Restore Only_Root command.

8.3.6 RMU Backup Operations Should Use Only One Type of Tape Drive

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

Oracle RMU attempts to prevent using different tape drive densities during a backup operation, but is not able to detect all invalid cases and expects that all tape drives for a backup are of the same type.

As long as all of the tapes used during a backup operation can be read by the same type of tape drive during a restore operation, the backup is likely valid. This may be the case, for example, when using a TA90 and a TA90E.

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

Consult the Oracle Rdb7 Guide to Database Maintenance, the Oracle Rdb7 Guide to Database Design and Definition, and the Oracle RMU Reference Manual for additional information about Oracle Rdb backup and restore operations.

8.3.7 RMU/VERIFY Reports PGSPAMENT or PGSPMCLST Errors

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

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

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

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

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

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

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

8.4 Known Problems and Restrictions in All Interfaces for Release 7.0 and Earlier

The following problems and restrictions from release 7.0 and earlier still exist.

8.4.1 Converting Single−File Databases

Because of a substantial increase in the database root file information for V7.0, you should ensure that you have adequate disk space before you use the RMU Convert command with single−file databases and V7.0 or higher.

The size of the database root file of any given database increases a 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.

8.4.2 Row Caches and Exclusive Access

If a table has a row−level cache defined for it, the Row Cache Server (RCS) may acquire a shared lock on the table and prevent any other user from acquiring a Protective or Exclusive lock on that table.

8.4.3 Exclusive Access Transactions May Deadlock with RCS Process

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

There are at least three suggested workarounds to this problem:

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

8.4.4 Strict Partitioning May Scan Extra Partitions

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

```
ATTACH 'FILENAME MF_PERSONNEL';
CREATE TABLE T1 (ID INTEGER, LAST_NAME CHAR(12), FIRST_NAME CHAR(12));
CREATE STORAGE MAP M FOR T1 PARTITIONING NOT UPDATABLE
 STORE USING (ID)
```

```
 IN EMPIDS_LOW WITH LIMIT OF (200)
 IN EMPIDS_MID WITH LIMIT OF (400)
 OTHERWISE IN EMPIDS_OVER;
INSERT INTO T1 VALUES (150,'Boney','MaryJean');
INSERT INTO T1 VALUES (350,'Morley','Steven');
INSERT INTO T1 VALUES (300,'Martinez','Nancy');
INSERT INTO T1 VALUES (450,'Gentile','Russ');
SELECT * FROM T1 WHERE ID > 400;
Conjunct Get Retrieval sequentially of relation T1
Strict Partitioning: part 2 3
ID LAST_NAME FIRST_NAME
450 Gentile Russ
1 row selected
```

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

8.4.5 Restriction When Adding Storage Areas with Users Attached to Database

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

```
%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
```

or

```
SQL>alter data file foo add storage area area6 page size 1 blocks;
%RDMS−F−NOEUACCESS, unable to acquire exclusive access to database
```

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

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

8.4.6 Multiblock Page Writes May Require Restore Operation

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

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

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

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

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

When a copy process completes updates to its respective remote database, it updates the RDB$TRANSFERS system relation and then tries to delete any RDB$CHANGES rows not needed by any transfers. During this process, the RDB$CHANGES table cannot be updated by any application process, holding up any database updates until the deletion process is complete. The application stalls while waiting for the RDB$CHANGES table. The resulting contention for RDB$CHANGES SPAM pages and data pages severely impacts performance throughput, requiring user intervention with normal processing.

This is a known restriction in V4.0 and higher. Oracle Rdb uses page locks as latches. These latches are held only for the duration of an action on the page and not to the end of transaction. The page locks also have blocking asynchronous system traps (ASTs) associated with them. Therefore, whenever a process requests a page lock, the process holding that page lock is sent a blocking AST (BLAST) by OpenVMS. The process that receives such a blocking AST queues the fact that the page lock should be released as soon as possible. However, the page lock cannot be released immediately.

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

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

8.5 SQL Known Problems and Restrictions for Oracle Rdb Release 7.0 and Earlier

The following problems and restrictions from Oracle Rdb Release 7.0 and earlier still exist.

8.5.1 ARITH_EXCEPT or Incorrect Results Using LIKE IGNORE CASE

When you use LIKE...IGNORE CASE, programs linked under Oracle Rdb V4.2 and V5.1, but run under higher versions of Oracle Rdb, may result in incorrect results or %RDB−E−ARITH_EXCEPT exceptions.

To work around the problem, avoid using IGNORE CASE with LIKE or recompile and relink under a higher version (V6.0 or higher.)

8.5.2 Different Methods of Limiting Returned Rows from Queries

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

If you define the RDMS$BIND_QG_REC_LIMIT logical name to a small value, the query often fails with no rows returned regardless of the value assigned to the logical. The following example demonstrates setting the limit to 10 rows and the resulting failure:

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

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

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

To see an indication of the queries executed against the system relations, define the RDMS$DEBUG_FLAGS logical name as "S" or "B".

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

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

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

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

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

8.5.3 Suggestions for Optimal Use of SHARED DATA DEFINITION Clause for Parallel Index Creation

The CREATE INDEX process involves the following steps:

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

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

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

- You should commit the transaction immediately after the CREATE INDEX statement so that locks on the table are released. This avoids lock conflicts with other index definers and improves overall concurrency.
- By assigning the location of the temporary sort work files SORTWORK0, SORTWORK1, ... , SORTWORK9 to different disks for each parallel process that issues the SHARED DATA DEFINITION statement, you can increase the efficiency of sort operations. This minimizes

any possible disk I/O bottlenecks and allows overlap of the SORT read/write cycle.

- If possible, enable global buffers and specify a buffer number large enough to hold a sufficient amount of table data. However, do not define global buffers larger than the available system physical memory. Global buffers allow sharing of database pages and thus result in disk I/O savings. That is, pages are read from disk by one of the processes and then shared by the other index definers for the same table, reducing the I/O load on the table.
- If global buffers are not used, ensure that enough local buffers exist to keep much of the index cached (use the RDM$BIND_BUFFERS logical name or the NUMBER OF BUFFERS IS clause in SQL to change the number of buffers).
- To distribute the disk I/O load, store the storage areas for the indexes on separate disk drives. Note that using the same storage area for multiple indexes results in contention during the index creation (Step 5) for SPAM pages.
- Consider placing the .ruj file for each parallel definer on its own disk or an infrequently used disk.
- Even though snapshot I/O should be minimal, consider disabling snapshots during parallel index creation.
- Refer to the Oracle Rdb7 Guide to Database Performance and Tuning to determine the appropriate working set values for each process to minimize excessive paging activity. In particular, avoid using working set parameters where the difference between WSQUOTA and WSEXTENT is large. The SORT utility uses the difference between these two values to allocate scratch virtual memory. A large difference (that is, the requested virtual memory grossly exceeds the available physical memory) may lead to excessive page faulting.
- The performance benefits of using SHARED DATA DEFINITION can best be observed when creating many indexes in parallel. The benefit is in the average elapsed time, not in CPU or I/O usage. For example, when two indexes are created in parallel using the SHARED DATA DEFINITION clause, the database must be attached twice, and the two attaches each use separate system resources.
- Using the SHARED DATA DEFINITION clause on a single−file database or for indexes defined in the RDB$SYSTEM storage area is not recommended.

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

Table 8−2 Elapsed Time for Index Creations

Index Create Job	Elapsed Time
Index1	00:02:22.50
Index2	00:01:57.94
Index3	00:02:06.27
Index4	00:01:34.53
Index5	00:01:51.96
Index6	00:01:27.57
Index7	00:02:34.64
Index8	00:01:40.56

Index9	00:01:34.43
Index10	00:01:47.44
All10	00:03:26.66

8.5.4 Side Effect When Calling Stored Routines

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

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

```
SQL> create module M
cont>     language SQL
cont>
cont>     procedure P (in :a integer, in :b integer, out :c integer);
cont>     begin
cont>     set :c = :a + :b;
cont>     end;
cont>
cont>     function F () returns integer
cont>     comment is 'expect F to always return 2';
cont>     begin
cont>     declare :b integer;
cont>     call P (1, 1, :b);
cont>     trace 'returning ', :b;
cont>     return :b;
cont>     end;
cont> end module;
SQL>
SQL> set flags 'TRACE';
SQL> begin
cont> declare :cc integer;
cont> call P (2, F(), :cc);
cont> trace 'Expected 4, got ', :cc;
cont> end;
~Xt: returning 2
~Xt: Expected 4, got 3
```

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

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

```
SQL> begin
cont> declare :bb, :cc integer;
cont> set :bb = F();
cont> call P (2, :bb, :cc);
cont> trace 'Expected 4, got ', :cc;
cont> end;
~Xt: returning 2
```

```
~Xt: Expected 4, got 4
```

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

8.5.5 Considerations When Using Holdable Cursors

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

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

- If the access strategy forces Oracle Rdb to take a data snapshot, the data read and cached may be stale by the time the cursor fetches the data.
  For example, user 1 opens a cursor and commits the transaction. User 2 deletes rows read by user 1 (this is possible because the read locks are released). It is possible for user 1 to report data now deleted and committed.
- If the access strategy uses indexes that allow duplicates, updates to the duplicates chain may cause rows to be skipped, or even revisited.
  Oracle Rdb keeps track of the dbkey in the duplicate chain pointing to the data that was fetched. However, the duplicates chain could be revised by the time Oracle Rdb returns to using it.

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

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

8.5.6 AIJSERVER Privileges

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

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

- ALTPRI
  This privilege allows the AIJSERVER to adjust its own priority to ensure adequate quorum (CPU utilization) to prompt message processing.
- PSWAPM
  This privilege allows the AIJSERVER to enable and disable process swapping, also necessary

to ensure prompt message processing.

- SETPRV
  This privilege allows the AIJSERVER to temporarily set any additional privileges it may need to access the standby database or its server processes.
- SYSPRV
  This privilege allows the AIJSERVER to access the standby database rootfile, if necessary.
- WORLD
  This privilege allows the AIJSERVER to more accurately detect standby database server process failure and handle network failure more reliably.

| Contents