Oracle® Rdb for OpenVMS

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Oracle® Rdb for OpenVMS

Release Notes

Release 7.0.8.2

September 2005

Oracle Rdb Release Notes, Release 7.0.8.2 for OpenVMS

Contents

Preface

Purpose of This Manual

This manual contains release notes for Oracle Rdb Release 7.0.8.2. The notes describe changed and enhanced features; upgrade and compatibility information; new and existing software problems and restrictions; and software and documentation corrections. These release notes cover both Oracle Rdb for OpenVMS Alpha and Oracle Rdb for OpenVMS VAX, which are referred to by their abbreviated name, Oracle Rdb.

Intended Audience

This manual is intended for use by all Oracle Rdb users. Read this manual before you install, upgrade, or use Oracle Rdb Release 7.0.8.2.

Document Structure

This manual consists of eight chapters:

Chapter 1	Describes how to install Oracle Rdb Release 7.0.8.2.
Chapter 2	Describes software errors corrected in Oracle Rdb Release 7.0.8.2.
Chapter 3	Describes software errors corrected in Oracle Rdb Release 7.0.8.1.
Chapter 4	Describes enhancements introduced in Oracle Rdb Release 7.0.8.1.
Chapter 5	Describes enhancements introduced in Oracle Rdb Release 7.0.8.
Chapter 6	Describes enhancements introduced in releases prior to Oracle Rdb Release 7.0.8.
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 7.0.8.2.

Chapter 1
Installing Oracle Rdb Release 7.0.8.2

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

NOTE

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

1.1 Requirements

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

- Oracle Rdb must be shutdown before you install this update kit. That is, the command file SYS$STARTUP:RMONSTOP(70).COM should be executed before proceeding with this installation. If you have an OpenVMS cluster, you must shutdown all versions of Oracle Rdb on all nodes in the cluster before proceeding.
- The installation requires approximately 130,000 free blocks on your system disk for OpenVMS VAX systems; 240,000 blocks for OpenVMS Alpha systems.

1.2 Invoking VMSINSTAL

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

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

variant−name

The variant names for the software update for Oracle Rdb Release 7.0.8.2 are:

- RDBSE2H070 for Oracle Rdb for OpenVMS VAX standard version.
- RDBASE2H070 for Oracle Rdb for OpenVMS Alpha standard version.
- RDBMVE2H070 for Oracle Rdb for OpenVMS VAX multiversion.
- RDBAMVE2H070 for Oracle Rdb for OpenVMS Alpha multiversion.

device−name

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

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

  ```
  DKA400:[RDBAMVE2H070.KIT]
  ```
- If the device is a magnetic tape drive, you need to specify only the device name. For example:

  ```
  MTA0:
  ```

OPTIONS N

This parameter prints the release notes.

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

```
$ @SYS$UPDATE:VMSINSTAL RDBAMVE2H070 MTA0: OPTIONS N
```

1.3 Stopping the Installation

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

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

1.4 After Installing Oracle Rdb

This update provides a new Oracle Rdb Oracle TRACE facility definition. Any Oracle TRACE selections that reference Oracle Rdb will need to be redefined to reflect the new facility version number for the updated Oracle Rdb facility definition, "RDBVMSV7.0−82".

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

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

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

```
$ LIBRARY /TEXT /EXTRACT=RDBVMSV7.0−82 −
_$ /OUT=RDBVMS.EPC$DEF SYS$SHARE:EPC$FACILITY.TLB
```

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

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

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

1.5 Spurious SYSVERDIF Message During Installation

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

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

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

1.6 Maximum OpenVMS Version Check Added

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

OpenVMS Version 8.2−x is the maximum supported version of OpenVMS.

As of Oracle Rdb Release 7.0.3, the Alpha EV6 processor is supported. As of Oracle Rdb Release 7.0.5, the Alpha EV67 processor is supported. As of Oracle Rdb Release 7.0.6, the Alpha Wildfire processor is supported (see http://metalink.oracle.com for specifics on which Wildfire configurations are supported). As of Oracle Rdb Release 7.0.6.2, the Alpha EV68 processor is supported. As of Oracle Rdb Release 7.0.7, the Alpha EV7 processor is supported.

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

Chapter 2
Software Errors Fixed in Oracle Rdb Release 7.0.8.2

This chapter describes software errors that are fixed by Oracle Rdb Release 7.0.8.2.

2.1 Software Errors Fixed That Apply to All Interfaces

2.1.1 Journals Not Initialized After Backup if Backing Up to Tape Device

Bug 2808539

If after image journal backups were being done to tape and the AIJs being backed up were circular AIJs, the backed up journal would not get properly initialized. This could lead to various issues such as journaling being shutdown with a "journal is not empty" error. When that occurred, the journal state displayed by RMU/DUMP/HEADER=JOURNAL would show the following lines of output:

```
          File is inaccessible
            journal has been made inaccessible by system
            journal is not empty
```

Other symptoms were also possible. The database recovery process (DBR) could fail with a bugcheck in DBR$RECOVER_ALL. Attempts to recover a database using an existing journal that was not properly re−initialized could fail with AIJCORRUPT errors. For example:

```
%RMU−W−AIJCORRUPT, journal entry 1451795/3364123 contains a new AIJBL that
doesn't have the start flag set
```

This problem was introduced in Oracle Rdb Release 7.0.7.3.

The problem can be avoided by backing up the journals to a disk destination.

This problem has been corrected in Oracle Rdb Release 7.0.8.2.

2.1.2 Wrong Result From UNION Query With Outer Join Leg

Bug 4392374

The following query should find one row.

```
set flags 'strategy,detail';
SELECT sec_id, busdate, tc_code
FROM (SELECT
        SP_sec_id,
        SP_busdate,
        SP_tc_code,
        SP_rlc_code,
        SP_stc_id,
        SP_part_id
FROM
    (SELECT
    SEC.sec_id,
    SEC.sec_sub_id,
    ABD.busdate,
```

```
STD.tc_code,
STD.rlc_code,
STD.stc_id,
OPA.part_id
FROM    STD    STD,
        ABD    ABD,
        OPA    OPA,
        RLC    RLC,
        STC    STC,
        SEC    SEC
WHERE   SEC.es_date   <=  ABD.busdate
AND     SEC.ee_date   >=  ABD.busdate
AND     SEC.inst_code =   1
AND     STD.es_date   <=  ABD.busdate
AND     STD.ee_date   >=  ABD.busdate
AND     STD.sec_id    =   SEC.sec_id
AND     OPA.sec_id    =   SEC.sec_id
AND     STD.rlc_code  =   RLC.rlc_code
AND     RLC.es_date   <=  ABD.busdate
AND     RLC.ee_date   >=  ABD.busdate
AND     STD.stc_id    =   STC.stc_id
AND     STC.es_date   <=  ABD.busdate
AND     STC.ee_date   >=  ABD.busdate
) AS    SP
    (SP_sec_id,
    SP_sec_sub_id,
    SP_busdate,
    SP_tc_code,
    SP_rlc_code,
    SP_stc_id,
    SP_part_id
    )
                                            LEFT OUTER JOIN
(SELECT
 DIV1.shr_id,
 DIV1.ex_div_date,
 DIV1.div_cur_code,
 SUM(DIV1.div_value)
 FROM
 DIV    DIV1,
 ABD    ABD1,
 SEC    SEC1,
 STD    STD1
 WHERE  DIV1.ex_div_date   =  ABD1.busdate
 AND    DIV1.div_cur_code  =  STD1.tc_code
 AND    STD1.es_date       <= ABD1.busdate
 AND    STD1.ee_date       >= ABD1.busdate
 AND    SEC1.sec_sub_id    =  DIV1.shr_id
 AND    STD1.sec_id        =  SEC1.sec_id
 AND    SEC1.es_date       <= ABD1.busdate
 AND    SEC1.ee_date       >= ABD1.busdate
 AND    SEC1.inst_code     =  1
 GROUP BY
 DIV1.shr_id,
 DIV1.ex_div_date,
 DIV1.div_cur_code
) AS    DIVSEC
        (DIVSEC_shr_id,
         DIVSEC_ex_div_date,
         DIVSEC_div_cur_code,
         DIVSEC_div_value)
ON      SP_sec_sub_id         =   DIVSEC_shr_id AND
```

```
            DIVSEC_div_cur_code   =   SP_tc_code AND
            DIVSEC_ex_div_date    =   SP_busdate
UNION ALL
    SELECT
            SEC.sec_id,
            ABD.busdate,
            STD.tc_code,
            STD.rlc_code,
            STD.stc_id,
            OPA.part_id
    FROM    STD    STD,
            SEC    SEC,
            OPA    OPA,
            ABD    ABD,
            RLC    RLC,
            STC    STC
    WHERE   SEC.es_date   <=  ABD.busdate
    AND     SEC.ee_date   >=  ABD.busdate
    AND     SEC.inst_code <>  1
    AND     STD.es_date   <=  ABD.busdate
    AND     STD.ee_date   >=  ABD.busdate
    AND     STD.sec_id    =   SEC.sec_id
    AND     OPA.sec_id    =   SEC.sec_id
    AND     STD.rlc_code  =   RLC.rlc_code
    AND     RLC.es_date   <=  ABD.busdate
    AND     RLC.ee_date   >=  ABD.busdate
    AND     STD.stc_id    =   STC.stc_id
    AND     STC.es_date   <=  ABD.busdate
    AND     STC.ee_date   >=  ABD.busdate
    )
    as OM_STATIC_VIEW (
    sec_id,
    busdate,
    tc_code,
    rlc_code,
    stc_id,
    part_id
    )
WHERE sec_id = 32978 AND busdate = '24-MAY-2005';
Tables:
  0 = STD
  1 = ABD
  2 = OPA
  3 = RLC
  4 = STC
  5 = SEC
  6 = DIV
  7 = ABD
  8 = SEC
  9 = STD
  10 = STD
  11 = SEC
  12 = OPA
  13 = ABD
  14 = RLC
  15 = STC
Conjunct: (<mapped field> = 32978) AND
          (<mapped field> = '24-MAY-2005')
Merge of 1 entries
  Merge block entry 1
  Merge of 2 entries
    Merge block entry 1
```

```
Conjunct: 5.sec_id = 32978
Conjunct: 1.busdate = '24-MAY-2005'
Conjunct: 5.sec_id = 32978          <== See Note1
Conjunct: 1.busdate = '24-MAY-2005'
Match    (Left Outer Join)                  <== See Note2
  Outer loop
    Sort: 1.busdate(a), 0.tc_code(a),
          5.sec_sub_id(a)          <== See Note3
    Merge of 1 entries
      Merge block entry 1
      Cross block of 6 entries
        Cross block entry 1
          Conjunct: 1.busdate = '24-MAY-2005'
          Index only retrieval of relation 1:ABD
            Index name  ABD_U_PRM [1:1]    Direct lookup
              Keys: <mapped field> = '24-MAY-2005'
        Cross block entry 2
          Conjunct: 4.es_date <= 1.busdate
          Conjunct: 4.ee_date >= 1.busdate
          Get     Retrieval by index of relation 4:STC
            Index name  STC_U_PRM [0:0]
        Cross block entry 3
          Conjunct: 0.es_date <= 1.busdate
          Conjunct: 0.ee_date >= 1.busdate
          Get
          Retrieval by index of relation 0:STD
            Index name  STD_U_FRG_7 [1:1]
              Keys: 0.stc_id = 4.stc_id
        Cross block entry 4
          Conjunct: 5.ee_date >= 1.busdate
          Get     Retrieval by index of relation 5:SEC   <== See Note2
            Index name  SEC_U_FRG_7 [2:3]
              Keys: (5.inst_code = 1) AND (0.sec_id =
                    5.sec_id) AND (5.es_date <=
                    1.busdate)
              Bool: 5.sec_id = 32978
        Cross block entry 5
          Conjunct: 3.ee_date >= 1.busdate
          Get     Retrieval by index of relation 3:RLC
            Index name  RLC_U_PRM [1:2]
              Keys: (0.rlc_code = 3.rlc_code)
                     AND (3.es_date <= 1.busdate)
        Cross block entry 6
          Get     Retrieval by index of relation 2:OPA
            Index name  OPA_U_PRM [1:1]    Direct lookup
              Keys: 2.sec_id = 5.sec_id
  Inner loop
    Temporary relation
    Sort: 6.ex_div_date(a), 6.div_cur_code(a), 6.shr_id(a)
    Merge of 1 entries
      Merge block entry 1
      Aggregate: 0:SUM (6.div_value)
      Sort: 6.shr_id(a), 6.ex_div_date(a), 6.div_cur_code
            (a)
      Cross block of 4 entries
        Cross block entry 1
          Index only retrieval of relation 7:ABD
            Index name  ABD_U_PRM [0:0]
        Cross block entry 2
          Get     Retrieval by index of relation 6:DIV
            Index name  DIV_U_ALT_1 [1:1]
              Keys: 6.ex_div_date = 7.busdate
```

```
            Cross block entry 3
              Conjunct: 8.ee_date >= 7.busdate
              Conjunct: 8.inst_code = 1
              Get     Retrieval by index of relation 8:SEC
                Index name  SEC_U_FRG_8 [1:2]
                  Keys: (8.sec_sub_id = 6.shr_id) AND (
                        8.es_date <= 7.busdate)
            Cross block entry 4
              Conjunct: (6.div_cur_code = 9.tc_code)
                         AND (9.ee_date >= 7.busdate)
              Get
              Retrieval by index of relation 9:STD
                Index name  STD_U_PRM [1:2]
                  Keys: (9.sec_id = 8.sec_id) AND (
                        9.es_date <= 7.busdate)
    Merge block entry 2
    Cross block of 6 entries
      Cross block entry 1
        Conjunct: 13.busdate = '24-MAY-2005'
        Index only retrieval of relation 13:ABD
          Index name  ABD_U_PRM [1:1]    Direct lookup
            Keys: <mapped field> = '24-MAY-2005'
      Cross block entry 2
        Conjunct: 15.es_date <= 13.busdate
        Conjunct: 15.ee_date >= 13.busdate
        Get     Retrieval by index of relation 15:STC
          Index name  STC_U_PRM [0:0]
      Cross block entry 3
        Conjunct: 10.es_date <= 13.busdate
        Conjunct: 10.ee_date >= 13.busdate
        Get     Retrieval by index of relation 10:STD
          Index name  STD_U_FRG_7 [1:1]
            Keys: 10.stc_id = 15.stc_id
      Cross block entry 4
        Conjunct: 14.ee_date >= 13.busdate
        Get     Retrieval by index of relation 14:RLC
          Index name  RLC_U_PRM [1:2]
            Keys: (10.rlc_code = 14.rlc_code) AND
                  (14.es_date <= 13.busdate)
      Cross block entry 5
        Conjunct: (11.ee_date >= 13.busdate) AND (
                  11.inst_code <> 1)
        Get     Retrieval by index of relation 11:SEC
          Index name  SEC_U_PRM [1:2]
            Keys: (10.sec_id = 11.sec_id) AND (11.es_date
                   <= 13.busdate)
            Bool: 11.sec_id = 32978
      Cross block entry 6
        Get     Retrieval by index of relation 12:OPA
          Index name  OPA_U_PRM [1:1]    Direct lookup
            Keys: 12.sec_id = 11.sec_id
0 rows selected
```

Note1:: The conjunct "5.sec_id = 32978" is generated from the predicate "sec_id = 32978" since sec_id is a mapped column from the OM_STATIC_VIEW (a derived table) which contains the UNION query with the first leg as a left outer join and the second leg as a regular join subquery.

Note2:: The match strategy is used for the left outer join operation and thus it requires a sort node. The sort operation in Rdb always pre−loaded the records from the table into the sort buffer before the actual record retrieval.

Note3:: When the conjunct "5.sec_id = 32978" is evaluated, the column "5.sec_id" contains a stale value from the table 5:SEC instead of the updated value from the sort buffer. This causes the query to return FALSE even though the correct record is found.

As a workaround, the query works if the index SEC_U_FRG_7 for SEC table is dropped since the query now switches from match to cross strategy for the left outer join operation without the sort node. This could also happen if the SQL flag is set to 'index_column_group' or 'old_cost_model'.

The following is the strategy output of the same query after the index SEC_U_FRG_7 is dropped.

```
Conjunct: (<mapped field> = 32978) AND
          (<mapped field> = '24-MAY-2005')
Merge of 1 entries
  Merge block entry 1
  Merge of 2 entries
    Merge block entry 1
    Conjunct: 5.sec_id = 32978                    <== See Note4
    Conjunct: 1.busdate = '24-MAY-2005'
    Cross block of 2 entries        (Left Outer Join)  <== See Note4
      Cross block entry 1
        Merge of 1 entries
          Merge block entry 1
          Cross block of 6 entries
            Cross block entry 1
              Conjunct: 1.busdate = '24-MAY-2005'
              Index only retrieval of relation 1:ABD
                Index name  ABD_U_PRM [1:1]    Direct lookup
                  Keys: <mapped field> = '24-MAY-2005'
            Cross block entry 2
              Conjunct: (5.ee_date >= 1.busdate) AND (
                        5.inst_code = 1)
              Get     Retrieval by index of relation 5:SEC  <== See Note4
                Index name  SEC_U_PRM [1:2]
                  Keys: (<mapped field> = 32978) AND (5.es_date <=
                        1.busdate)
            Cross block entry 3
              Conjunct: 4.es_date <= 1.busdate
              Conjunct: 4.ee_date >= 1.busdate
              Get     Retrieval by index of relation 4:STC
                Index name  STC_U_PRM [0:0]
            Cross block entry 4
              Conjunct: 0.ee_date >= 1.busdate
              Conjunct: 0.stc_id = 4.stc_id
              Get
              Retrieval by index of relation 0:STD
                Index name  STD_U_PRM [1:2]
                  Keys: (0.sec_id = 5.sec_id) AND (
                        0.es_date <= 1.busdate)
            Cross block entry 5
              Get     Retrieval by index of relation 2:OPA
                Index name  OPA_U_PRM [1:1]    Direct lookup
                  Keys: 2.sec_id = 5.sec_id
            Cross block entry 6
              Conjunct: 3.ee_date >= 1.busdate
              Get     Retrieval by index of relation 3:RLC
                Index name  RLC_U_PRM [1:2]
                  Keys: (0.rlc_code = 3.rlc_code)
                         AND (3.es_date <= 1.busdate)
      Cross block entry 2
        Conjunct: (5.sec_sub_id = 6.shr_id) AND (
```

```
                6.div_cur_code = 0.tc_code) AND (
                6.ex_div_date = 1.busdate)
      Merge of 1 entries
        Merge block entry 1
        Aggregate: 0:SUM (6.div_value)
        Sort: 6.shr_id(a), 6.ex_div_date(a), 6.div_cur_code
              (a)
        Cross block of 4 entries
          Cross block entry 1
            Index only retrieval of relation 7:ABD
              Index name  ABD_U_PRM [0:0]
          Cross block entry 2
            Conjunct: (5.sec_sub_id = 6.shr_id) AND (
                      6.div_cur_code = 0.tc_code) AND
                      (6.ex_div_date = 1.busdate)
            Conjunct: 6.ex_div_date = 7.busdate
            Get     Retrieval by index of relation 6:DIV
              Index name  DIV_U_ALT_1 [2:2]
                Keys: (6.ex_div_date = 7.busdate) AND (
                      5.sec_sub_id = 6.shr_id)
          Cross block entry 3
            Conjunct: 8.ee_date >= 7.busdate
            Conjunct: 8.inst_code = 1
            Get     Retrieval by index of relation 8:SEC
              Index name  SEC_U_FRG_8 [1:2]
                Keys: (8.sec_sub_id = 6.shr_id) AND (
                      8.es_date <= 7.busdate)
          Cross block entry 4
            Conjunct: (6.div_cur_code = 9.tc_code)
                       AND (9.ee_date >= 7.busdate)
            Get
            Retrieval by index of relation 9:STD
              Index name  STD_U_PRM [1:2]
                Keys: (9.sec_id = 8.sec_id) AND (
                      9.es_date <= 7.busdate)
  Merge block entry 2
  Cross block of 6 entries
    Cross block entry 1
      Conjunct: 13.busdate = '24-MAY-2005'
      Index only retrieval of relation 13:ABD
        Index name  ABD_U_PRM [1:1]    Direct lookup
          Keys: <mapped field> = '24-MAY-2005'
    Cross block entry 2
      Conjunct: 15.es_date <= 13.busdate
      Conjunct: 15.ee_date >= 13.busdate
      Get     Retrieval by index of relation 15:STC
        Index name  STC_U_PRM [0:0]
    Cross block entry 3
      Conjunct: 10.es_date <= 13.busdate
      Conjunct: 10.ee_date >= 13.busdate
      Get     Retrieval by index of relation 10:STD
        Index name  STD_U_FRG_7 [1:1]
          Keys: 10.stc_id = 15.stc_id
    Cross block entry 4
      Conjunct: 14.ee_date >= 13.busdate
      Get     Retrieval by index of relation 14:RLC
        Index name  RLC_U_PRM [1:2]
          Keys: (10.rlc_code = 14.rlc_code) AND
                (14.es_date <= 13.busdate)
    Cross block entry 5
      Conjunct: (11.ee_date >= 13.busdate) AND (
                11.inst_code <> 1)
```

```
        Get     Retrieval by index of relation 11:SEC
          Index name  SEC_U_PRM [1:2]
            Keys: (10.sec_id = 11.sec_id) AND (11.es_date
                  <= 13.busdate)
            Bool: 11.sec_id = 32978
      Cross block entry 6
        Get     Retrieval by index of relation 12:OPA
          Index name  OPA_U_PRM [1:1]    Direct lookup
            Keys: 12.sec_id = 11.sec_id
 sec_id   busdate                   tc_code
  32978   24-MAY-2005 00:00:00.00   CHF
1 row selected
```

Note4:: The cross strategy is used for the left outer join operation instead of a match strategy. Since no sort is required and the column 5.sec_id in the conjunct "5.sec_id = 32978" did not get pre−loaded as in the case of a sort operation, it was correctly retrieved from the table on the fly during the cross operation.

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

1. The main query selects a particular row from either a view or derived table of a union query where the first leg is a left outer join subquery and the second leg is a regular join subquery.
2. The column of the equality predicate is mapped via the view or derived table to the same base table which is included in both legs of the left outer join subquery and the second UNION leg.
3. The query applies a match strategy for the left outer join operation with a sort node at the outer loop and another sort node at the inner loop.

This problem has been corrected in Oracle Rdb Release 7.0.8.2.

2.1.3 COSI_MEM_FREE_VMLIST Bugcheck with Vertical Partitioning

Bug 4460398

When vertical partitioning was in use, the following bugcheck was possible.

```
***** Exception at 0091B704 : COSI_MEM_FREE_VMLIST + 00000094
%SYSTEM-F-ACCVIO, access violation, reason mask=04,
  virtual address=000000000008308D, PC=000000000091B704, PS=0000001B
```

This problem has been corrected in Oracle Rdb Release 7.0.8.2.

2.1.4 Bugcheck from INSERT with Partition Index

Bug 4477862

The following query bugchecks.

```
SQL>INSERT INTO TBL1 VALUES (1,2,3);
%DEBUG-I-DYNMODSET, setting module RDMS$PREEXEMSC
%SYSTEM-F-ACCVIO, access violation, reason mask=00, virtual address=000000000000
0060, PC=00000000006A2358, PS=00000018
1816:                           IF .CXPR [CXPR$L_OPERATOR] NEQU BLR$K_MISSING
```

```
where the tables are defined as follows:

CREATE TABLE TBL1 (TBL1_P1 INT, TBL1_P2 INT, TBL1_P3 INT, CONSTRAINT TBL1_PRIM
PRIMARY KEY (TBL1_P1, TBL1_P2) DEFER);

CREATE TABLE TBL2 (TBL2_P1 INT, TBL2_P2 INT, TBL2_P3 INT, CONSTRAINT TBL2_FOR
FOREIGN KEY (TBL2_P1, TBL2_P2) REFERENCES TBL1 (TBL1_P1, TBL1_P2) DEFER);

CREATE INDEX IDX1 ON TBL1 (TBL1_P1 DESC, TBL1_P2 DESC) STORE USING (TBL1_P1)
IN A1 WITH LIMIT OF (2)
IN A2 WITH LIMIT OF (1)
OTHERWISE IN A3;
CREATE INDEX IDX2 ON TBL2 (TBL2_P1, TBL2_P2) ;
COMMIT;

UPDATE RDB$RELATIONS SET RDB$CARDINALITY = 306
WHERE RDB$RELATION_NAME='TBL2';
COMMIT;
```

As a workaround, the query works if the cardinality of table TBL2 is updated to a number below 306 rows.

```
UPDATE RDB$RELATIONS SET RDB$CARDINALITY = 305
WHERE RDB$RELATION_NAME='TBL2';
COMMIT;
```

This problem in the INSERT statement occurs when the following conditions are met:

1. The table TBL1 contains a primary constraint and the table TBL2 contains a foreign key constraint referencing the primary keys of TBL1.
2. The index for the table TBL1 is a partition index keying on the leading primary keys.
3. The cardinality of the table TBL2 is updated to a number such that the optimizer chooses a match strategy over cross. In this case, the cardinality number is 306.

This problem has been corrected in Oracle Rdb Release 7.0.8.2.

2.1.5 Wrong Result from Constant View Column

Bug 4448141

The following query should find one row.

```
set flags 'strategy,detail';
select employee_id, i.const_col, j.const_col
from employees
     join mfp_v1 i using (employee_id)
     left outer join mfp_v2 j using (employee_id)
     where i.const_col = 9 and j.const_col is NULL
Tables:
  0 = EMPLOYEES
  1 = EMPLOYEES
  2 = SALARY_HISTORY
  3 = EMPLOYEES
  4 = SALARY_HISTORY
Conjunct: (9 = 9) AND MISSING (9)
Match    (Left Outer Join)
  Outer loop
```

```
    Conjunct: 0.EMPLOYEE_ID = 1.EMPLOYEE_ID
    Match
      Outer loop      (zig-zag)
        Index only retrieval of relation 0:EMPLOYEES
          Index name  EMP_EMPLOYEE_ID [0:0]
      Inner loop
        Temporary relation
        Cross block of 2 entries
          Cross block entry 1
            Index only retrieval of relation 1:EMPLOYEES
              Index name  EMP_EMPLOYEE_ID [0:0]
          Cross block entry 2
            Conjunct: <agg0> <> 0
            Aggregate-F1: 0:COUNT-ANY (<subselect>)
            Index only retrieval of relation 2:SALARY_HISTORY
              Index name  SH_EMPLOYEE_ID [1:1]
                Keys: 1.EMPLOYEE_ID = 2.EMPLOYEE_ID
  Inner loop
    Temporary relation
    Sort: 3.POSTAL_CODE(a)
    Conjunct: <agg1> <> 0
    Match    (Agg Outer Join)
      Outer loop
        Get     Retrieval by index of relation 3:EMPLOYEES
          Index name  EMP_EMPLOYEE_ID [0:0]
      Inner loop      (zig-zag)
        Aggregate-F1: 1:COUNT-ANY (<subselect>)
        Index only retrieval of relation 4:SALARY_HISTORY
          Index name  SH_EMPLOYEE_ID [0:0]
0 rows selected

where the views mfp_v1 and mfp_v2 are defined as follows:

create view mfp_v1 (employee_id, const_col) as
select a.employee_id, 9 from employees a
  where a.employee_id = any (select z.employee_id from salary_history z )
  ;

create view mfp_v2 (employee_id, const_col) as
select a.postal_code, 9 from employees a
  where a.employee_id = any (select z.employee_id from salary_history z )
  ;
```

If the WHERE predicates are removed, the query finds 100 rows with the values 9 and NULL, as in the following example.

```
select employee_id, i.const_col, j.const_col
from employees
     join mfp_v1 i using (employee_id)
     left outer join mfp_v2 j using (employee_id);
Tables:
  0 = EMPLOYEES
  1 = EMPLOYEES
  2 = SALARY_HISTORY
  3 = EMPLOYEES
  4 = SALARY_HISTORY
Match    (Left Outer Join)
  Outer loop
    Conjunct: 0.EMPLOYEE_ID = 1.EMPLOYEE_ID
    Match
      Outer loop      (zig-zag)
```

```
        Index only retrieval of relation 0:EMPLOYEES
          Index name  EMP_EMPLOYEE_ID [0:0]
      Inner loop
        Temporary relation
        Cross block of 2 entries
          Cross block entry 1
            Index only retrieval of relation 1:EMPLOYEES
              Index name  EMP_EMPLOYEE_ID [0:0]
          Cross block entry 2
            Conjunct: <agg0> <> 0
            Aggregate-F1: 0:COUNT-ANY (<subselect>)
            Index only retrieval of relation 2:SALARY_HISTORY
              Index name  SH_EMPLOYEE_ID [1:1]
                Keys: 1.EMPLOYEE_ID = 2.EMPLOYEE_ID
  Inner loop
    Temporary relation
    Sort: 3.POSTAL_CODE(a)
    Conjunct: <agg1> <> 0
    Match    (Agg Outer Join)
      Outer loop
        Get     Retrieval by index of relation 3:EMPLOYEES
          Index name  EMP_EMPLOYEE_ID [0:0]
      Inner loop      (zig-zag)
        Aggregate-F1: 1:COUNT-ANY (<subselect>)
        Index only retrieval of relation 4:SALARY_HISTORY
          Index name  SH_EMPLOYEE_ID [0:0]
 EMPLOYEE_ID   I.CONST_COL   J.CONST_COL
 00164                   9          NULL
 00165                   9          NULL
 00166                   9          NULL
 00167                   9          NULL
...etc...
 00435                   9          NULL
 00471                   9          NULL
100 rows selected
```

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 two views left outer joined by the common column 'employee_id' where both views contain the constant literal as one of the columns. In this case, the constant literal is '9'.
2. The filter predicates in the WHERE clause reference the constant column of each view.

This problem has been corrected in Oracle Rdb Release 7.0.8.2.

2.1.6 Wrong Result from UNION View Query with NOT STARTING WITH Clause

Bugs 4550715 and 4327112

The following query should select 2 rows from the FW_V view for rows where column F3 does not start with 'Z'. As indicated next to the rows returned, the results are wrong. This is due to an incorrect query strategy, as will be explained further on.

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

```
SELECT * FROM FW_V WHERE F3 NOT STARTING WITH 'Z';
Tables:
  0 = TAB_A
  1 = TAB_B
  2 = TAB_B
  3 = TAB_C
Reduce: <mapped field>, <mapped field>, <mapped field>
Sort: <mapped field>(a), <mapped field>(a), <mapped field>(a)
Merge of 2 entries
  Merge block entry 1
  Cross block of 3 entries
    Cross block entry 1
      Get Retrieval sequentially of relation 0:TAB_A
    Cross block entry 2
      Aggregate: 0:VIA (1.FLD3)
      Conjunct: 1.FLD2 = 0.FLD2
      Get     Retrieval sequentially of relation 1:TAB_B
    Cross block entry 3
      Aggregate: 1:VIA (2.FLD3)
      Conjunct: 2.FLD2 = 0.FLD2
      Get     Retrieval sequentially of relation 2:TAB_B
  Merge block entry 2
  Conjunct: NOT (3.F3 STARTING WITH 'Z')
  Get     Retrieval sequentially of relation 3:TAB_C
 FLD1   FLD2   F3
 1      A      Z      <== wrong
 2      A      Z      <== wrong
 3      B      1      <== correct
 4      C      Z      <== wrong
 aaa    bbb    ccc    <== correct
5 rows selected
```

The view, FW_V, is defined below. Following the view definition is a SELECT query similar to the problem query shown earlier but without the clause "WHERE F3 NOT STARTING WITH 'Z'".

```
CREATE VIEW FW_V AS
SELECT
    FLD1,
    A.FLD2,
    NVL((SELECT FLD3 FROM TAB_B B WHERE B.FLD2 = A.FLD2), 'Z')
    FROM TAB_A A
UNION
SELECT F1, F2, F3
    FROM TAB_C;

select * from fw_v;
 FLD1   FLD2   F3
 1      A      Z
 2      A      Z
 3      B      1
 4      C      Z
 aaa    bbb    ccc
5 rows selected
```

The query returns the correct result if the NOT operator is removed and the reason is that the conjunct "STARTING WITH 'Z'" appears correctly in the detailed strategy, as in the following example.

```
SELECT * FROM FW_V WHERE F3  STARTING WITH 'Z';
Tables:
  0 = TAB_A
```

```
  1 = TAB_B
  2 = TAB_B
  3 = TAB_C
Reduce: <mapped field>, <mapped field>, <mapped field>
Sort: <mapped field>(a), <mapped field>(a), <mapped field>(a)
Conjunct: <mapped field> STARTING WITH 'Z'  <=== This conjunct appears here
Merge of 2 entries
  Merge block entry 1
  Cross block of 3 entries
    Cross block entry 1
      Get     Retrieval sequentially of relation 0:TAB_A
    Cross block entry 2
      Aggregate: 0:VIA (1.FLD3)
      Conjunct: 1.FLD2 = 0.FLD2
      Get     Retrieval sequentially of relation 1:TAB_B
    Cross block entry 3
      Aggregate: 1:VIA (2.FLD3)
      Conjunct: 2.FLD2 = 0.FLD2
      Get     Retrieval sequentially of relation 2:TAB_B
  Merge block entry 2
  Conjunct: 3.F3 STARTING WITH 'Z'
  Get     Retrieval sequentially of relation 3:TAB_C
 FLD1   FLD2   F3
 1      A      Z
 2      A      Z
 4      C      Z
3 rows selected
```

If we compare the strategies of the good and bad queries, the only difference is that the bad query was missing the conjunct "NOT STARTING WITH", but the second good query (the one without the NOT operator), correctly places the conjunct "STARTING WITH 'Z'" outside the "Merge" entries.

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 the derived table of a UNION clause with a filter predicate.
2. The first leg of the UNION clause contains a select query with a NVL function on a subselect query.
3. The second leg of the UNION clause contains a select query from a table.
4. The WHERE clause contains NOT operator in front of a string function, e.g. STARTING WITH.

This problem has been corrected in Oracle Rdb Release 7.0.8.2.

2.1.7 Unexpected Bugcheck when Formatting Illegal Date/Time Value

Bug 4557990

In previous versions of Oracle Rdb, it was possible that an invalid value in a TIME, DATE (ansi), or TIMESTAMP would result in a reported ACCVIO or a BUGCHECK from Rdb. This might occur when using Interactive SQL to display the value of when using RMU/UNLOAD to format the value as text.

This problem has been corrected in Oracle Rdb Release 7.0.8.2. Oracle Rdb now uses a field of *** to indicate the illegal date/time data.

2.1.8 Wrong Result by Query with Constant Column Defined in a View

Bugs 1752645 and 4155086

The following query should returns 100 rows with NULL for the column "v.const_col".

```
create view mfp_view (employee_id, const_col) as
select e.postal_code, 1 from employees e
  where e.employee_id = any (select z.employee_id from salary_history z );

select employee_id, v.const_col
  from employees left outer join mfp_view v using (employee_id)
where v.const_col is null;
Tables:
  0 = EMPLOYEES
  1 = EMPLOYEES
  2 = SALARY_HISTORY
Conjunct: MISSING (1)
Match    (Left Outer Join)
  Outer loop
    Index only retrieval of relation 0:EMPLOYEES
      Index name  EMP_EMPLOYEE_ID [0:0]
  Inner loop
    Temporary relation
    Sort: 1.POSTAL_CODE(a)
    Conjunct: <agg0> <> 0
    Match    (Agg Outer Join)
      Outer loop
        Get     Retrieval by index of relation 1:EMPLOYEES
          Index name  EMP_EMPLOYEE_ID [0:0]
      Inner loop      (zig-zag)
        Aggregate-F1: 0:COUNT-ANY (<subselect>)
        Index only retrieval of relation 2:SALARY_HISTORY
          Index name  SH_EMPLOYEE_ID [0:0]
0 rows selected
```

This problem occurs when the main query is an outer join between the EMPLOYEES table and a view using the EMPLOYEE_ID as the join predicate and the view contains a column of either constant value or an expression of all constant values.

```
create view mfp_view (employee_id, const_col) as
select e.postal_code, 1+2 from employees e
  where e.employee_id = any (select z.employee_id from salary_history z );
select employee_id, v.const_col
  from employees left outer join mfp_view v using (employee_id)
where v.const_col is null;
Tables:
  0 = EMPLOYEES
  1 = EMPLOYEES
  2 = SALARY_HISTORY
Conjunct: MISSING (1 + 2)
Match    (Left Outer Join)
  Outer loop
    Index only retrieval of relation 0:EMPLOYEES
      Index name  EMP_EMPLOYEE_ID [0:0]
  Inner loop
    Temporary relation
```

```
    Sort: 1.POSTAL_CODE(a)
    Conjunct: <agg0> <> 0
    Match    (Agg Outer Join)
      Outer loop
        Get     Retrieval by index of relation 1:EMPLOYEES
          Index name  EMP_EMPLOYEE_ID [0:0]
      Inner loop      (zig-zag)
        Aggregate-F1: 0:COUNT-ANY (<subselect>)
        Index only retrieval of relation 2:SALARY_HISTORY
          Index name  SH_EMPLOYEE_ID [0:0]
0 rows selected
```

A similar query works if the constant column in the view is concatenated with another column of the EMPLOYEES table (for example middle_initial).

```
create view mfp_view (employee_id, const_col) as
select e.postal_code, '1'||e.middle_initial from employees e
  where e.employee_id = any (select z.employee_id from salary_history z );

select employee_id, v.const_col
  from employees left outer join mfp_view v using (employee_id)
where v.const_col is null;
Tables:
  0 = EMPLOYEES
  1 = EMPLOYEES
  2 = SALARY_HISTORY
Conjunct: MISSING ('1' || 1.MIDDLE_INITIAL)
Match    (Left Outer Join)
  Outer loop
    Index only retrieval of relation 0:EMPLOYEES
      Index name  EMP_EMPLOYEE_ID [0:0]
  Inner loop
    Temporary relation
    Sort: 1.POSTAL_CODE(a)
    Conjunct: <agg0> <> 0
    Match    (Agg Outer Join)
      Outer loop
        Get     Retrieval by index of relation 1:EMPLOYEES
          Index name  EMP_EMPLOYEE_ID [0:0]
      Inner loop      (zig-zag)
        Aggregate-F1: 0:COUNT-ANY (<subselect>)
        Index only retrieval of relation 2:SALARY_HISTORY
          Index name  SH_EMPLOYEE_ID [0:0]
 EMPLOYEE_ID   I.CONST_COL
 00164         NULL
 00165         NULL
  ...etc...
 00471         NULL
100 rows selected
```

The fix for this bug was backed out to fix the problem for Bug 4155086. It is now redesigned with the correct solution which also correctly handles the case where a view constant column of the same value exists in more than one view.

The following example shows that a view constant of the value '9' exists in two different views.

```
attach 'file mf_personnel';

create view mfp_v1 (employee_id, const_col) as
select a.employee_id, 9 from employees a
```

```
  where a.employee_id = any (select z.employee_id from salary_history z )
  group by a.employee_id;


create view mfp_v2 (employee_id, const_col) as
select a.postal_code, 9 from employees a
  where a.employee_id = any (select z.employee_id from salary_history z )
  group by a.postal_code;

The following query should find 100 rows with values 9 and NULL but
found only 0 row :

select count(*)
from employees
     join mfp_v1 i using (employee_id)
     left outer join mfp_v2 j using (employee_id)
     where i.const_col = 9 and j.const_col is NULL
      ;
Aggregate       Conjunct
Match    (Left Outer Join)
  Outer loop
    Conjunct
    Match
      Outer loop
        Reduce
        Cross block of 2 entries
          Cross block entry 1
            Index only retrieval of relation EMPLOYEES
              Index name  EMP_EMPLOYEE_ID [0:0]
          Cross block entry 2
            Conjunct        Aggregate-F1
            Index only retrieval of relation SALARY_HISTORY
              Index name  SH_EMPLOYEE_ID [1:1]
      Inner loop      (zig-zag)
        Index only retrieval of relation EMPLOYEES
          Index name  EMP_EMPLOYEE_ID [0:0]
  Inner loop
    Temporary relation      Reduce  Sort    Conjunct
    Match    (Agg Outer Join)
      Outer loop
        Get     Retrieval by index of relation EMPLOYEES
          Index name  EMP_EMPLOYEE_ID [0:0]
      Inner loop      (zig-zag)
        Aggregate-F1    Index only retrieval of relation SALARY_HISTORY
          Index name  SH_EMPLOYEE_ID [0:0]

           0
1 row selected
```

There is no known workaround for this problem.

This problem has been corrected in Oracle Rdb Release 7.0.8.2.

2.1.9 Bugchecks or Corruption in Indexes of TYPE IS SORTED RANKED

Bug 4437323

During a sequence of inserts on a table with an index of *TYPE IS SORTED RANKED*, it was possible that the index could become corrupt. This could cause the index to be left corrupt or could result in a bugcheck dump if subsequent delete operations were performed in the same transaction.

The problem is quite rare and requires that a series of at least three consecutive inserts occur on exactly the same duplicate node in a specific pattern.

If the index was corrupted, a subsequent verify would produce output similar to the following example.

```
SQL> insert into t1 values (0,33);
1 row inserted
SQL> insert into t1 values (0,11);
1 row inserted
SQL> insert into t1 values (0,22);
1 row inserted
SQL> commit;
SQL>  Exit
$ rmu/verify/index/data test
%RMU-W-IDXDATMIS, Index I1 does not point to a row in table T1.
                  Logical dbkey of the missing row is 68:12:1.
%RMU-W-BADIDXREL, Index I1 either points to a non-existent record or
                  has multiple pointers to a record in table T1.
                  The logical dbkey in the index is 68:14:1.
```

Various bugchecks may result depending on how the corruption was detected. The following shows two different examples of the call frames in bugcheck dumps from this problem.

```
***** Exception at 012D7DC8 : PIOFETCH$WITHIN_DB + 000005A8
Saved PC = 012D5414 : PIOFETCH$FETCH + 000002C4
Saved PC = 012D44D4 : PIO$FETCH + 00000914
Saved PC = 01220044 : DIOFETCH$FETCH_VISIBLE_SEG + 00000154
Saved PC = 01221414 : DIOFETCH$FETCH_ONE_LINE + 00000AB4
Saved PC = 01221978 : DIO$FETCH_DBKEY + 000002F8
Saved PC = 01044848 : RDMS$$KOD_FIND_CURRENT_REC + 00000558
Saved PC = 01023684 : RDMS$$EXE_NEXT + 00000954

***** Exception at 0141D9DC : PSII2REMOVEDUPBBC + 0000118C
Saved PC = 0141A3CC : PSII2REMOVEBOTTOM + 0000071C
Saved PC = 01413E74 : PSII2REMOVET + 000001F4
Saved PC = 014140B4 : PSII2REMOVET + 00000434
Saved PC = 014140B4 : PSII2REMOVET + 00000434
Saved PC = 014140B4 : PSII2REMOVET + 00000434
Saved PC = 0141481C : PSII2REMOVETREE + 000001FC
Saved PC = 016DA80C : RDMS$$KOD_REMOVE_TREE + 0000162C
Saved PC = 016A29DC : RDMS$$EXE_ACTION + 0000321C
```

If the bugcheck occurred during the transaction that corrupted the index, the transaction would be rolled back and the index would not be left corrupt.

If the problem did not cause a bugcheck during the offending transaction, the transaction could commit and the index would be left corrupt. The index should be verified using the */INDEX/DATA* qualifiers of RMU VERIFY. If errors are reported, the index should be dropped and recreated to eliminate the corruption.

The problem can be avoided by adding columns to the index to make it unique, or by using alternate index types.

This problem has been corrected in Oracle Rdb Release 7.0.8.2.

2.2 Oracle RMU Errors Fixed

2.2.1 RMU /UNLOAD Output File Maximum Record Size

Bug 4573066

Previously, the RMU /UNLOAD command, when writing in TEXT format, would sometimes incorrectly set the output file maximum record size field in the file header.

This problem has been corrected in Oracle Rdb Release 7.0.8.2.

2.3 LogMiner Errors Fixed

2.3.1 RMU /UNLOAD /AFTER_JOURNAL Field Order Clarification

Unlike SQL and RMU /UNLOAD, the Oracle Rdb LogMiner (tm) RMU /UNLOAD /AFTER_JOURNAL command outputs fields in the database on−disk field order. This difference can become apparant when a table definition has been modified. For example, when adding a new column to appear in a positon other than at the end of the record.

The following example shows one way that the Oracle Rdb LogMiner (tm) RMU /UNLOAD /AFTER_JOURNAL command outputs fields in the database on−disk field order.

```
$!
$ SQL$
    CREATE DATA FILE FOO LOGMINER SUPPORT ENA;
    -- Create table then insert another column between COL1 & COL2.

    CREATE TABLE T1 (COL1 INT, COL2 INT);
    ALTER TABLE T1 ADD COLUMN COL3 INT BEFORE COLUMN COL2;
    SHOW TABLE (COLUMN) T1;
Information for table T1

Columns for table T1:
Column Name                     Data Type        Domain
-----------                     ---------        ------
COL1                            INTEGER
COL3                            INTEGER
COL2                            INTEGER

    COMMIT;

    DISCONNECT ALL;
    ALTER DATA FILE FOO JOU ENA ADD JOU J1 FILE J1;
    EXIT;
$!
$ RMU/BACKUP/NOLOG FOO NLA0:FOO
$ RMU/BACKUP/AFTER/NOLOG FOO NLA0:B1
$!
$ SQL$
    ATTACH 'FILE FOO';
    INSERT INTO T1 VALUES (NULL,2,3); -- COL1=NULL, COL3=2, COL2=3
1 row inserted
    INSERT INTO T1 VALUES (1,NULL,3); -- COL1=1, COL3=NULL, COL2=3
1 row inserted
    INSERT INTO T1 VALUES (1,2,NULL); -- COL1=1, COL3=2, COL2=NULL
1 row inserted
    COMMIT;
    SELECT * FROM T1; -- Show data content
        COL1          COL3          COL2
        NULL             2             3
           1          NULL             3
           1             2          NULL
3 rows selected
    EXIT;
$!
```

```
$ RMU/BACKUP/AFTER/NOLOG FOO B1
$!
$ RMU/UNL/RECORD=FILE=T1.RRD1 FOO T1 NLA0:T1
%RMU-I-DATRECUNL,   3 data records unloaded.
$ RMU/UNL/AFTER/NOLOG/FORMAT=DUMP FOO B1 -
    /TABLE=(NAME=T1,OUTPUT=T1.DAT,RECORD=T1.RRD2)
$!
$ SEARCH T1.RRD1 COL ! Show field order - RMU/UNLOAD
DEFINE FIELD COL1 DATATYPE IS SIGNED LONGWORD.
DEFINE FIELD COL3 DATATYPE IS SIGNED LONGWORD.
DEFINE FIELD COL2 DATATYPE IS SIGNED LONGWORD.
   COL1 .
   COL3 .
   COL2 .
$ SEARCH T1.RRD2 COL ! Show field order - RMU/UNLOAD/AFTER_JOURNAL
DEFINE FIELD COL1 DATATYPE IS SIGNED LONGWORD.
DEFINE FIELD COL2 DATATYPE IS SIGNED LONGWORD.
DEFINE FIELD COL3 DATATYPE IS SIGNED LONGWORD.
   COL1.
   COL2.
   COL3.
$ SEARCH T1.DAT RDB$LM_RELATION_NAME, COL ! Show data content
RDB$LM_RELATION_NAME            : T1
COL1                            : NULL
COL2                            : 3
COL3                            : 2
RDB$LM_RELATION_NAME            : T1
COL1                            : 1
COL2                            : 3
COL3                            : NULL
RDB$LM_RELATION_NAME            : T1
COL1                            : 1
COL2                            : NULL
COL3                            : 2
$!
```

Chapter 3
Software Errors Fixed in Oracle Rdb Release 7.0.8.1

This chapter describes software errors that are fixed by Oracle Rdb Release 7.0.8.1.

3.1 Software Errors Fixed That Apply to All Interfaces

3.1.1 Problem with Rdb 7.0.8 and VMS Versions Below V7.1

It was discovered that Rdb Release 7.0.8 would not install on any VMS versions below V7.1 (ie V5.5−2) due to some changes that had been made in the software.

This problem has been corrected in Oracle Rdb Release 7.0.8.1. Rdb Release 7.0.8.1 will install and run successfully on VMS versions as low as V5.5−2.

3.1.2 Wrong Results Generated by Query With Common Boolean Elements

Bug 4332115

In prior releases of Oracle Rdb, an optimization was applied to WHERE clauses and other Boolean expressions to reformat those queries to gain possible advantages in the query execution phase. However, in the reported problem, this optimization leads to a query strategy that did not return the correct results. In some cases, the common expression in an OR expression was lifted too high in the expression and so distorted the results.

While this problem is possible in older versions of Rdb, it may occur more frequently in Rdb V7.0 and later versions because of a more aggressive restructure algorithm employed by these recent versions.

For example, this query against the EMPLOYEES table should produce four result rows.

```
SQL> set flags 'strategy,detail';
SQL>
SQL> select last_name, first_name
cont> from employees
cont> where
cont>     (
cont>      (
cont>       (last_name = 'Watters' and first_name = 'Christine')
cont>       or
cont>       (last_name = 'Watters' and first_name = 'Cora')
cont>      )
cont>      and
cont>      (
cont>       (last_name = 'Watters' and first_name = 'Christine')
cont>       or
cont>       (last_name = 'Watters' and first_name = 'Cora')
cont>      )
cont>     )
cont>     or
cont>     (last_name = 'Smith')
cont> order by 1, 2
cont> ;
Tables:
  0 = EMPLOYEES
```

```
Sort: 0.LAST_NAME(a), 0.FIRST_NAME(a)
Leaf#01 BgrOnly 0:EMPLOYEES Card=100
  Bool: ((((0.FIRST_NAME = 'Christine') OR (0.FIRST_NAME = 'Cora')) AND ((
        0.FIRST_NAME = 'Christine') OR (0.FIRST_NAME = 'Cora'))) OR (0.LAST_NAME
         = 'Smith')) AND (0.LAST_NAME = 'Watters')
  BgrNdx1 EMP_LAST_NAME [1:1] Fan=12
    Keys: 0.LAST_NAME = 'Watters'
 LAST_NAME        FIRST_NAME
 Watters          Christine
 Watters          Cora
2 rows selected
SQL> -- expecting 4 rows
```

The detailed strategy output shows that the expression *AND (0.LAST_NAME = 'Watters')* had been raised to the outer most part of the query and thus erroneously eliminated two of the rows matching 'Smith'.

This problem has been corrected in Oracle Rdb Release 7.0.8.1. The query optimizer now correctly handles the case where a trailing OR term does not match a common Boolean with the query.

3.1.3 Query With Shared Expressions in OR Predicates Returns Wrong Result

Bugs 4300529 and 3918278

The following query with shared expressions in an OR predicate returns the wrong result.

```
select * from
  (SELECT T1.SEM, T1.PAL, T1.VUO, T1.KK, T1.PV
     FROM T1, T2
     WHERE (T2.SEM = T1.SEM)
          AND
          ((NOT EXISTS (SELECT T3.SEM FROM T3
                        WHERE T3.SEM = T1.SEM))
           OR
           (T1.SEM = 'JOVK'))
           AND
          ((NOT EXISTS (SELECT T3.SEM FROM T3
                        WHERE T3.PAL = T1.PAL))
           OR
           (T1.SEM = 'JOVK'))
    ) as DTAB (SEM, PAL, VUO, KK, PV)
where VUO = '2005' and KK = '03' and PV = '29';
Tables:
  0 = T1
  1 = T2
  2 = T3
  3 = T3
Merge of 1 entries
  Merge block entry 1
  Cross block of 4 entries
    Cross block entry 1
      Conjunct: (0.VUO = '2005') AND (0.KK = '03') AND (0.PV = '29')
      Index only retrieval of relation 0:T1
        Index name  T1_INDEX [3:3]
          Keys: (0.VUO = '2005') AND (0.KK = '03') AND (0.Pv = '29')
    Cross block entry 2
      Conjunct: (<agg0> = 0) OR (0.SEM = 'JOVK')
      Conjunct: (<agg0> = 0) OR (0.SEM = 'JOVK')
```

```
      Aggregate-F1: 0:COUNT-ANY (<subselect>)
      Conjunct: (2.SEM = 0.SEM)
      Index only retrieval of relation 2:T3
        Index name  T3_INDEX [0:0]
    Cross block entry 3
      Conjunct: (<agg0> = 0) OR (0.SEM = 'JOVK')
      Conjunct: 0.SEM = 'JOVK'                    <== See NOTE
      Conjunct: ((<agg0> = 0) OR (0.SEM = 'JOVK')) AND ((<agg1> = 0) OR
                (0.SEM = 'JOVK'))
      Aggregate-F1: 1:COUNT-ANY (<subselect>)
      Conjunct: (<agg0> = 0) OR (0.SEM = 'JOVK')
      Conjunct: (<agg0> = 0) OR (0.SEM = 'JOVK')
      Conjunct: (3.PAL = 0.PAL)
      Index only retrieval of relation 3:T3
        Index name  T3_INDEX [0:0]
    Cross block entry 4
      Conjunct: ((<agg0> = 0) OR (0.SEM = 'JOVK')) AND ((<agg1> = 0) OR
                (0.SEM = 'JOVK'))
      Index only retrieval of relation 1:T2
        Index name  T2_INDEX [1:1]     Direct lookup
          Keys: 1.SEM = 0.SEM
 SEM    PAL          VUO    KK     PV
 JOVK   TV2          2005   03     29
 JOVK   TV2          2005   03     29
2 rows selected
```

NOTE:: The conjunct "(0.SEM = 'JOVK')" is separated from its parent OR predicate with the other left operand "(<agg0> = 0)".

The following cross entry 3 contains the incorrect conjunct "(0.SEM = 'JOVK')" which is separated from the other left operand "(<agg0> = 0)" of the OR predicate.

```
    Cross block entry 3
      Conjunct: (<agg0> = 0) OR (0.SEM = 'JOVK')
      Conjunct: 0.SEM = 'JOVK'                     <== Incorrect
      Conjunct: ((<agg0> = 0) OR (0.SEM = 'JOVK')) AND ((<agg1> = 0) OR
                (0.SEM = 'JOVK'))
      Aggregate-F1: 1:COUNT-ANY (<subselect>)
      Conjunct: (<agg0> = 0) OR (0.SEM = 'JOVK')
      Conjunct: (<agg0> = 0) OR (0.SEM = 'JOVK')
      Conjunct: (3.PAL = 0.PAL)
      Index only retrieval of relation 3:T3
        Index name  T3_INDEX [0:0]
```

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 derived table joining two tables with the filtering predicates in the outer WHERE clause.
2. The inner WHERE clause of the query selecting the derived table contains a join equality predicate and two similar OR predicates with a shared expression "(T1.SEM = 'JOVK')" as the right side operand.

This problem has been corrected in Oracle Rdb Release 7.0.8.1.

3.1.4 Various Errors or Corruption of Ranked Indexes

Bug 4216643

If a series of updates occurred on an index node of *TYPE IS SORTED RANKED* in the same transaction, it was possible that the index node could be corrupted. Several different results could occur depending on subsequent operations in the transaction.

- If there were no further operations on the index node in the same transaction and the transaction committed, the index would be left corrupt.
- If a subsequent update attempted certain operations on the index node, various bugchecks could result. In this case, the transaction would be rolled back and the index node would not be corrupt.
- If a subsequent update attempted to update the same index node, the update could fail with the error message "RDB−E−NO_RECORD, access by dbkey failed because dbkey is no longer associated with a record". In this case, the index might or might not be left corrupt depending on the precise sequence of operations and also on the application's response to the error. If the application commited the transaction, the index might be left corrupt. If the application rolled back the transaction, the index would not be left corrupt.

In the reported case, a RDB−E−NO_RECORD error was reported, and the failed update was automatically rolled back. This is termed a *verb rollback*. The following example shows the reported error.

```
SQL>delete from some_table where some_key < 20050225;
%RDB-E-NO_RECORD, access by dbkey failed because dbkey is no longer
associated with a record
-RDMS-F-NODBK, 98:770:1 does not point to a data record
```

In this case, a subsequent *RMU/VERIFY* reported no errors and the index was not corrupt.

To produce this problem, a precise sequence of operations needed to occur within the same transaction.

- A row is deleted where the key value is not the first key value in the index node and that key value has many duplicates causing the duplicates chain to overflow into several overflow index nodes.
- The dbkey for the deleted row must be in the first overflow node for the key value.
- A row is deleted that is the last remaining row with a key value in the same index node and that key value appeared in the index node before the previously deleted row.
- Another row is deleted with the first key value and the deleted dbkey is the last remaining dbkey in the first overflow node.

When the last dbkey is deleted from the first overflow node, that node is deleted and the overflow pointer in the index node must be modified to point to the second overflow node. In the sequence above, the deletion of the unique key values caused the location of the second ikey to be moved in the index node but the third delete used a stale pointer to update the overflow dbkey.

The problem can be avoided by performing the sequence of operations in a different order or in separate transactions. The problem only affects indexes of *type is sorted ranked*.

If index corruption occurs, the index must be dropped and recreated to eliminate the corruption.

This problem has been corrected in Oracle Rdb Release 7.0.8.1.

3.1.5 Wrong Result From Query With Common Join Booleans in OR

Bugs 4332115 and 1329838

The following query with common join booleans in an OR predicate returns the wrong result (should be 5 rows) after the common boolean optimization is manually disabled (by commenting out the code).

```
sel  e.employee_id, e.last_name, j.job_start
  from employees e, job_history j
  where
        e.employee_id = j.employee_id and e.employee_id = '00222'
              or
        e.employee_id = j.employee_id and e.employee_id = '00234'
              or
        e.employee_id = j.employee_id and e.employee_id = '00345'
;
Tables:
  0 = EMPLOYEES
  1 = JOB_HISTORY
Cross block of 2 entries
  Cross block entry 1
    Get     Retrieval by index of relation 0:EMPLOYEES
      Index name  EMP_EMPLOYEE_ID [0:0]
  Cross block entry 2
    Conjunct:
    ((0.EMPLOYEE_ID = 1.EMPLOYEE_ID) AND (0.EMPLOYEE_ID = '00222'))
              OR
    ((0.EMPLOYEE_ID = 1.EMPLOYEE_ID) AND (0.EMPLOYEE_ID = '00234'))
              OR
    ((0.EMPLOYEE_ID = 1.EMPLOYEE_ID) AND (0.EMPLOYEE_ID = '00345'))
    OR index retrieval
      Conjunct:
      ((0.EMPLOYEE_ID = 1.EMPLOYEE_ID) AND (0.EMPLOYEE_ID = '00222'))
                OR
      ((0.EMPLOYEE_ID = 1.EMPLOYEE_ID) AND (0.EMPLOYEE_ID = '00234'))
      OR index retrieval
        Conjunct:
        (0.EMPLOYEE_ID = 1.EMPLOYEE_ID) AND (0.EMPLOYEE_ID = '00222')
        Get     Retrieval by index of relation 1:JOB_HISTORY
          Index name  JOB_HISTORY_HASH [1:1]
            Keys: 0.EMPLOYEE_ID = 1.EMPLOYEE_ID
        Conjunct:
        NOT ((0.EMPLOYEE_ID = 1.EMPLOYEE_ID) AND (0.EMPLOYEE_ID = '00222'))
              AND
        (0.EMPLOYEE_ID = 1.EMPLOYEE_ID) AND (0.EMPLOYEE_ID = '00234')
        Get     Retrieval by index of relation 1:JOB_HISTORY
          Index name  JOB_HISTORY_HASH [1:1]
            Keys: 0.EMPLOYEE_ID = 1.EMPLOYEE_ID
      Conjunct:
      NOT (((0.EMPLOYEE_ID = 1.EMPLOYEE_ID) AND (0.EMPLOYEE_ID = '00222'))
              OR
           (0.EMPLOYEE_ID = 1.EMPLOYEE_ID) )   ! Note :: <== missing '00234'
      AND (0.EMPLOYEE_ID = 1.EMPLOYEE_ID) AND (0.EMPLOYEE_ID = '00345')
      Get     Retrieval by index of relation 1:JOB_HISTORY
        Index name  JOB_HISTORY_HASH [1:1]
          Keys: 0.EMPLOYEE_ID = 1.EMPLOYEE_ID
 E.EMPLOYEE_ID   E.LAST_NAME      J.JOB_START
 00222           Lasch            28-Dec-1979
```

```
 00222           Lasch           18-Aug-1976
 00234           Robinson        20-May-1980
 00234           Robinson         5-Mar-1978
4 rows selected
```

Note that the boolean '0.EMPLOYEE_ID = '00234' is missing in the NOT predicate of the 2nd leg of the outer OR index retrieval.

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 joins EMPLOYEES and JOB_HISTORY tables using EMPLOYEE_ID.
2. The WHERE clause of the query contains two OR predicates with three operands where E.EMPLOYEE_ID = J.EMPLOYEE_ID is a common join boolean in each operand.

This problem has been corrected in Oracle Rdb Release 7.0.8.1.

3.1.6 Wrong Result Selecting From a Derived Table of UNION Clause

Bug 4327112

The following query selects the wrong result (3 rows) from a derived table of a union clause.

```
set flags 'strategy,detail';
SEL * FROM
    (SELECT FLD1, A.FLD2,
        NVL((SELECT FLD3 FROM TAB_B B WHERE B.FLD2 = A.FLD2), 'Z')
     FROM TAB_A A
     UNION
     SELECT F1, F2, F3 FROM TAB_C
     ) AS DT (F1, F2, F3)
WHERE DT.F3 = 'Z';
Tables:
  0 = TAB_A
  1 = TAB_B
  2 = TAB_C
Merge of 1 entries
  Merge block entry 1
  Reduce: <mapped field>, <mapped field>, <mapped field>
  Sort: <mapped field>(a), <mapped field>(a), <mapped field>(a)
  Merge of 2 entries
    Merge block entry 1
    Cross block of 2 entries
      Cross block entry 1
        Get Retrieval sequentially of relation 0:TAB_A
      Cross block entry 2
        Aggregate: 0:VIA (1.FLD3)
        Conjunct: 1.FLD2 = 0.FLD2
        Get     Retrieval sequentially of relation 1:TAB_B
    Merge block entry 2
    Conjunct: 2.F3 = 'Z'
    Get     Retrieval sequentially of relation 2:TAB_C
 F1     F2     F3
```

```
 1      A      Z
 2      A      Z
 3      B      1      <== should not have returned this row
 4      C      Z
4 rows selected
```

In the problem query, the conjunct "DT.F3 = 'Z'" appears only in the second UNION leg but not in the first UNION leg. Without an additional conjunct at the outside of the union query, the query returns the wrong result.

As a workaround, the query works if the union legs are swapped.

```
SEL * FROM
    (
    SELECT F1, F2, F3 FROM TAB_C ! <== second leg is swapped here as first leg
    UNION
    SELECT
        FLD1, A.FLD2,
        NVL((SELECT FLD3 FROM TAB_B B WHERE B.FLD2 = A.FLD2), 'Z')
        FROM TAB_A A
        ) AS DT (F1, F2, F3)
WHERE DT.F3 = 'Z';
Tables:
  0 = TAB_C
  1 = TAB_A
  2 = TAB_B
  3 = TAB_B
Conjunct: <mapped field> = 'Z'                <== See note
Merge of 1 entries
  Merge block entry 1
  Reduce: <mapped field>, <mapped field>, <mapped field>
  Sort: <mapped field>(a), <mapped field>(a), <mapped field>(a)
  Merge of 2 entries
    Merge block entry 1
    Conjunct: 0.F3 = 'Z'
    Get     Retrieval sequentially of relation 0:TAB_C
    Merge block entry 2
    Cross block of 3 entries
      Cross block entry 1
        Get     Retrieval sequentially of relation 1:TAB_A
      Cross block entry 2
        Aggregate: 0:VIA (2.FLD3)
        Conjunct: 2.FLD2 = 1.FLD2
        Get     Retrieval sequentially of relation 2:TAB_B
      Cross block entry 3
        Aggregate: 1:VIA (3.FLD3)
        Conjunct: 3.FLD2 = 1.FLD2
        Get     Retrieval sequentially of relation 3:TAB_B
 F1     F2     F3
 1      A      Z
 2      A      Z
 4      C      Z
3 rows selected
```

Note:: There is an additional conjunct "<mapped field> = 'Z'" at the top of the union query.

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

1. The main query selects from the derived table of a union clause with a filter predicate.

2. The first leg of the union clause contains a select query with a NVL function on a subselect query.
3. The second leg of the union clause contains a select query from a table.

This problem has been corrected in Oracle Rdb Release 7.0.8.1.

3.1.7 Incorrect Foreign Key Constraint Behavior on Update

Bug 4157145

A certain class of foreign key constraint would fail to detect a violation when an update was performed. Under the following conditions, an update statement that modified a primary/unique key would not result in a constraint violation if one were to exist:

- The constraint must have been defined by the SQL REFERENCES clause (making it a foreign key constraint).
- The constraint must be self−referencing. A self−referencing constraint is one in which the columns of the foreign key and the columns of the primary/unique key are in the same table.
- The update statement must have referenced one or more of the columns that make up the primary/unique key.

The following is an example of a self−referencing, foreign key constraint.

```
$ SQL$

create database filename test;

create table t (pk char (3), fk char (3),
                constraint pk_constraint
                  primary key (pk) not deferrable);

insert into t(pk)    values ('1');
1 row inserted
insert into t(pk,fk) values ('2','2');
1 row inserted
insert into t(pk,fk) values ('3','1');
1 row inserted
commit;

alter table t
  add constraint
    constraint fk_constraint
      foreign key (fk) references t(pk) not deferrable;
commit;

select * from t order by pk;
 PK     FK
 1      NULL
 2      2
 3      1
3 rows selected
rollback;

update t set pk='9' where pk='1';
%RDB−E−INTEG_FAIL, violation of constraint FK_CONSTRAINT caused
operation to fail
−RDB−F−ON_DB, on database DISK:[DIR]TEST.RDB;
```

The third row, with values (3,1), shows a foreign key value that matches a primary key value in the first row. The update statement, which attempts to change the primary key value in that first row from 1 to 9 violates the foreign key constraint that a primary key with value 1 exist in the table.

The example shows the error message that is now reported by Oracle Rdb Release 7.0.8.1. In some prior releases, the constraint violation was not detected, no error message was reported, and the update was allowed.

For self−referencing, foreign key constraint definitions created using SQL and the REFERENCES clause and created using Oracle Rdb Release 7.0.8.1, the correct behavior will appear. For such constraints created by certain earlier releases of Oracle Rdb, the problem will continue to appear. To determine if a table contains erroneous data, data that should cause a constraint violation, execute the statement RMU/VERIFY/CONSTRAINT. If there is an error in the data, you should see an error message such as the one in the following example. In the example, the database file specification has been abbreviated to the word DBASE to shorten the text.

```
$       rmu/verify/constraint test.rdb
%RMU−I−BGNROOVER, beginning root verification
%RMU−I−ENDROOVER, completed root verification
%RMU−I−BGNVCONST, beginning verification of constraints for database DBASE
%RMU−W−CONSTFAIL, Verification of constraint "FK_CONSTRAINT" has failed.
%RMU−I−ENDVCONST, completed verification of constraints for database DBASE
...
```

Oracle recommends the regular use of RMU Verify to monitor the integrity of your databases.

To correct the problem in constraint definitions created using earlier releases of Oracle Rdb, first fix the database files, either by dropping and recreating the offending constraint definitions or by recreating the database files from EXPORT/IMPORT interchange files. Then, recreate any Oracle Rdb backup files for the databases (files with filename extensions of .RBF).

- For an existing Rdb database, alter the table definition to drop the foreign key constraint. Then, alter the table definition to recreate the constraint using Oracle Rdb Release 7.0.8.1.
- For an existing database backup file, make a new backup file from a database file in which the constraint definition has been recreated as explained in the preceding bulleted item.

There is no known workaround for this problem.

This problem has been corrected in Oracle Rdb Release 7.0.8.1.

3.1.8 Bugchecks in PSII2SPLITNODE When Using Ranked Indexes

Bug 4324725

When inserting rows into a table with indexes of *TYPE IS SORTED RANKED*, it was possible that a bugcheck could occur in the routine PSII2SPLITNODE. The exception occurred when the 65535th row was added to a particular key value and the dbkey was inserted into an overflow node and the level one node contained exactly one key value and had two or fewer free bytes.

The following example shows the bugcheck footprint for an index that only contains one key value.

```
COSI-F-BUGCHECK, internal consistency failure
Exception occurred at PSII2SPLITNODE + 00000390
Called from PSII2CASETOPM1 + 000006C8
Called from PSII2INSERTTREE + 000001FC
Called from RDMS$$KOD_INSERT_TREE + 00002954
```

For indexes with more than one key value, the bugcheck footprint would be slightly different.

```
COSI-F-BUGCHECK, internal consistency failure
Exception occurred at PSII2SPLITNODE + 00000390
Called from PSII2BALANCE + 00000DEC
Called from PSII2INSERTT + 00000548
Called from PSII2INSERTT + 0000042C
```

The index is not corrupt but repeated attempts to insert such a record will fail with the same exception. If the index is dropped, the row may be inserted, and the index can be rebuilt correctly. A rebuild of the index would likely eliminate the bugcheck.

The problem can be avoided by either using alternate index types or adding fields to the key value to make the index more unique.

This problem has been corrected in Oracle Rdb Release 7.0.8.1.

3.1.9 Connection Name Longer than 31 Characters Mishandled

Bug 4380993

If a connection name was submitted which had greater than the allowed length of 31 characters, the name would be silently truncated. This has been corrected so that the CONNECT statement is rejected with a %SQL-E-CONTOOBIG error.

In addition, if a SET CONNECT or DELETE CONNECT statement were submitted with a name larger than 31 characters, the error text associated with the returned %SQL-E-NOSUCHCON error sometimes contained garbage characters. This has been fixed so that the garbage characters will no longer appear.

As a workaround, use a connection name of 31 characters or less.

This problem has been corrected in Oracle Rdb Release 7.0.8.1.

3.2 SQL Errors Fixed

3.2.1 Dynamic SQL Rounds Results from Division Operator

Bug 4165206

In previous versions of Oracle Rdb, the Dynamic SQL interface would determine the result data type using the type and scale of the dividend. This lead to rounded results if the dividend was a fixed point value. With this release of Oracle Rdb, all numeric division computations return a REAL or DOUBLE PRECISION value.

The following example shows the behavior in prior versions.

```
$ r test$tools:tester
Enter statement:
attach 'filename sql$database';
Enter statement:
select 1/2 from rdb$database;
 0/: 1
Enter statement:
```

In this example the result (0.5) is rounded to the target type of INTEGER to yield a (possibly unexpected) value 1. Now Dynamic SQL uses a floating result and gives the expected result.

```
$ r test$tools:tester
Enter statement:
attach 'filename sql$database';
Enter statement:
select 1/2 from rdb$database;
 0/: 0.500000
Enter statement:
```

This problem has been corrected in Oracle Rdb Release 7.0.8.1.

3.3 LogMiner Errors Fixed

3.3.1 RMU /UNLOAD /AFTER_JOURNAL Incorrect Settings in Null Bit Vector

Bug 4367366

With some combinations of table definition modifications with adding and removing fields, it was possible for a table's maximum null bit vector length to be incorrectly calculated by the Oracle Rdb LogMiner(tm). The incorrect length would be used internally by the RMU /UNLOAD /AFTER_JOURNAL command and could result in an incorrect null bit vector content being generated for the output stream.

As a possible workaround, unloading the table data from the database, dropping and recreating the table and then reloading the content would cause the table's maximum field identification to be reset and the RMU /UNLOAD /AFTER_JOURNAL command would then work correctly.

This problem has been corrected in Oracle Rdb Release 7.0.8.1. The RMU /UNLOAD /AFTER_JOURNAL command now correctly determines the maximum field ID and the null bit vector length.

Chapter 4
Enhancements Provided in Oracle Rdb Release 7.0.8.1

4.1 Enhancements Provided in Oracle Rdb Release 7.0.8.1

4.1.1 New DEFAULTS Qualifier Added to RMU Extract

This release of Oracle Rdb adds a new DEFAULTS qualifier to RMU Extract.

- Defaults=defaults−list
  This qualifier is used to change the output of the RMU Extract command. The following defaults can be modified with the Defaults qualifier:
  - Allocation
    NoAllocation
    When creating a test database using the RMU Extract generated script, the allocation from the source database may not be appropriate. The ALLOCATION keyword can be used to specify an alternate value to be used by all storage areas or the NOALLOCATION keyword can be used to cause the clause to be omitted from the CREATE STORAGE MAP syntax. The default behavior, that is when neither keyword is used, is to use the allocation recorded in the database for each storage area. See also the SNAPSHOT_ALLOCATION keyword.
  - Date_format
    NoDate_format
    By default, RMU Extract assumes that DATE types will be SQL standard compliant (that is DATE ANSI) and that the builtin function CURRENT_TIMESTAMP will return a TIMESTAMP(2) value. If your environment uses DATE VMS exclusively, then you can modify the default by specifying the default DATE_FORMAT=VMS. The legal values are described in the Oracle Rdb SQL Reference Manual in the SET DEFAULT DATE FORMAT section. The default is DATE_FORMAT=SQL92.
    Use NODATE_FORMAT to omit the setting of this session attribute from the script.
  - Dialect
    NoDialect
    For some extracted SQL scripts, the language dialect is required to be specified. The DIALECT keyword can be used to supply a specified dialect for the script. The legal values for this option can be found in the Oracle Rdb SQL Reference Manual in the SET DIALECT section. The default is NODIALECT.
  - Language
    NoLanguage
    RMU Extract uses the process language, that is the translated value of SYS$LANGUAGE or ENGLISH, for the SET LANGUAGE command. However, if the script is used on a different system then this language might not be appropriate. The LANGUAGE keyword can be used to supply a specified language for the script. Legal language names are defined by the OpenVMS system logical name table. Examine the logical name SYS$LANGUAGES for the current setting. Use NOLANGUAGE to omit this command from the script.
  - Quoting_rules
    NoQuoting_rules
    The QUOTING_RULES keyword can be used to supply a specified setting for the script. The legal values for this option can be found in the Oracle Rdb SQL Reference Manual in the SET QUOTING RULES section. The default is QUOTING_RULES=SQL92. Please note that RMU Extract assumes that SQL keywords and names containing non−ASCII character set values will be quoted.

- Snapshot_allocation
  NoSnapshot_allocation
  When creating a test database from the RMU Extract output, the snapshot file allocation from the source database may not be appropriate. The SNAPSHOT_ALLOCATION keyword can be used to specify an alternate value to be used by all snapshot areas or the NOALLOCATION keyword can be used to cause the "snapshot allocation is" clause to be omitted. The default behavior (that is when neither keyword is used), is to use the snapshot allocation stored in the database for each snapshot area. See also the ALLOCATION keyword.

Chapter 5
Enhancements Provided in Oracle Rdb Release 7.0.8

5.1 Enhancements Provided in Oracle Rdb Release 7.0.8

5.1.1 Support for OpenVMS Version 8.2

This version of Rdb, Release 7.0.8, supports HP's OpenVMS Version 8.2 Release.

5.1.2 RDM$BIND_SNAP_QUIET_POINT Logical Reinstated

Bug 3908414

Over the years, various problems have been reported related to quiet point backups. In particular, database backups and journal backups would sometimes fail with the following error:

```
%RMU-F-TIMEOUT, timeout on quiet
```

Quiet point backups are required so that recovery of a journal can be done without always requiring previous journals. Full database backups can avoid lock conflicts if they wait for the quiet point lock. See the documentation for the RMU Backup commands for more information regarding quiet point backups.

Lock timeout errors for the quiet lock are intended to be returned when a long running update transaction has not completed within the timeout period. However, Oracle Rdb Release 6.0 forced all transactions (read−only or read−write) to obtain the quiet point lock when starting. That greatly increased the incidence of timeout errors from backups. To remedy this situation, a special logical name, RDM$BIND_SNAP_QUIET_POINT, was implemented that would force processes starting read−only transactions to release the quiet point lock. If that logical was defined to the value 0 then read−only transactions would not obtain the quiet point lock.

Defining the RDM$BIND_SNAP_QUIET_POINT logical to 0 would usually resolve problems with timeouts on the quiet lock, but it would effectively disable the fast commit performance feature for applications that often switched between read−only and read−write transactions. (See Bug 884004.) To remedy that situation, the transaction start code was modified to retain the quiet lock during read−only transactions but release the lock during the read−only transaction if a backup process started. With this change, the RDM$BIND_SNAP_QUIET_POINT lock logical could be defined without impacting the performance of the fast commit feature. That change was introduced in Releases 7.0.6.3 and 7.1.0.1.

The previous fix resolved performance problems, but the logical still could not be defined to 0 if the Hot Standby feature was being used. (See Bug 2656534.) If the Hot Standby feature was being utilized, the logical had to be undefined, or defined to the default value of 1. Applications that utilized Hot Standby were still subject to undeserved timeouts from backup commands. In Releases 7.0.7.1 and 7.1.2, the Hot Standby feature was modified so that read−only transactions were not required to hold the quiet point lock. Also, because read−only processes would usually release the quiet point lock when a backup started, the RDM$BIND_SNAP_QUIET_POINT logical was completely removed.

After the above changes, quiet lock timeouts were no longer an issue for most applications. However, a read−only transaction would only release the quiet point lock when Oracle Rdb had returned control to the user application. Some complicated queries could execute for an exceptionally long period of time before returning a row to the user application and thus might not release the quiet lock in time to prevent timeout

errors for the backup process. For example, read−only queries that executed aggregate functions such as SUM or AVERAGE on large sets of rows, or queries that required sorts of large sets of rows before any rows were returned could sometimes not release the quiet point lock before the backup command timed out. (See Bug 3908414.) In addition, any update transaction that started after the backup process requested the quiet point lock would stall until the long running read−only request returned control to the user application. That means that it was possible for many database users to stall waiting for the read−only transaction to complete, even though they had released the quiet point lock.

This release of Oracle Rdb reinstates the RDM$BIND_SNAP_QUIET_POINT logical so that read−only transactions can be forced to release the quiet point lock before starting. The logical now has a slightly different meaning than the original implementation. The default value is still 1, but that value now signifies that all transactions will hold the quiet point lock until a backup process requests it. Read−only transactions will not obtain the quiet point lock; only read−write requests will obtain the quiet point lock. This is the behavior that was introduced in response to Bug 884004. If the logical is defined to be 0, then read−only transactions will always release the quiet point lock at the beginning of the transaction, regardless of the existence of a backup process. That implies that all modified buffers in the buffer pool have to be written to disk before the transaction proceeds. Applications that utilize the fast commit feature and often switch between read−only and read−write transactions within a single attach may experience performance degradation if the logical is defined to 0. This is the behavior that was in place prior to Releases 7.0.6.3 and 7.1.0.1.

Oracle Corporation recommends that you do not define the RDM$BIND_SNAP_QUIET_POINT logical for most applications. If the scenario described in Bug 3908414 is being encountered, the logical can be defined to 0 to force read−only transactions to always release the quiet point lock. Rather than define the logical system−wide, this logical can be defined for specific jobs that are likely to execute for an extensive period of time before returning to the user application. This parameter may also be manipulated from the RMU Show Statistics locking dashboard. That way most users will not have to sacrifice fast commit performance when switching between read−only and read−write transactions. As of releases 7.0.7.1 and 7.1.2, Hot Standby is no longer affected by this logical so Hot Standby is not a factor when determining how to define the logical.

5.1.3 RMU Unload After_Journal/Ignore Old_Version Keyword

The RMU Unload After_Journal command treats non−current record versions in the AIJ file as a fatal error condition. That is, attempting to extract a record that has a record version not the same as the table's current maximum version results in a fatal error.

There are, however, some very rare cases where a verb rollback of a modification of a record may result in an old version of a record being written to the after−image journal even though the transaction did not actually complete a successful modification to the record. The RMU Unload After_Journal command detects the old record version and aborts with a fatal error in this unlikely case.

The RMU Unload After_Journal command now accepts a new keyword to partially work around this case. The Ignore qualifier has been enhanced to include the keyword Old_Version. When this keyword is present, the RMU Unload After_Journal command displays a warning message for each record that has a non−current record version and the record is not written to the output stream. The Old_Version keyword accepts an optional list of table names to indicate that only the specified tables are permitted to have non−current record version errors ignored.

5.1.4 New Features in RMU Extract

There have been several enhancements to the RMU Extract command. The following description of the command is the RMU Extract command chapter as it would appear in the *Oracle RMU Reference Manual* for Release 7.0.8.

RMU Extract Command

Reads and decodes Oracle Rdb metadata and reconstructs equivalent statements in Relational Database Operator (RDO) or SQL (structured query language) code for the definition of that database. These statements can either be displayed or extracted. You can use these statements to create your database again if you no longer have the RDO or SQL code that defined your database.

In addition, you can direct the RMU Extract command to produce output for the following:

- An SQL or RDO IMPORT script (Items=Import)
- An RMU Unload command for each table (Items=Unload)
- An RMU Load command for each table (Items=Load)
- An RMU Set Audit command for the database (Items=Security)
- An RMU Verify command for the database (Items=Verify)

RMU/Extract root-file-spec

Command Qualifiers	Defaults
/Items[=item-list]	/Items=All
/Language=lang-name	/Language=SQL
/[No]Log[=log-file]	/Nolog
/Options=options-list	/Option=Normal
/[No]Output[=out-file]	/Output=SYS$OUTPUT
/Transaction_Type[=(transation_mode,options...)]	See Description

---

DESCRIPTION

The RMU Extract command decodes information and reconstructs equivalent commands in the language you select with the Language qualifier for the definition of that database.

You can extract the definitions to either a file or to SYS$OUTPUT. The RMU Extract command extracts the following character set information:

- For databases:
  - ♦ The database default character set
  - ♦ The national character set
- For domains:

    - ♦ The character set of each character data type domain
    - ♦ The length in characters of each character data type domain
- For tables:
    - ♦ The character set of each character data type column
    - ♦ The length in characters of each character data type column

The RMU Extract command may enclose object names in double quotation marks to preserve the uppercase and lowercase characters, to represent different character sets, or to allow processing of SQL reserved keywords.

COMMAND PARAMETERS

root−file−spec

The file specification for the database root file from which you want to extract definitions. Note that you do not need to specify the file extension. If the database root file is not found, the command exits with a "file not found" error.

COMMAND QUALIFIERS

Items[=item−list]

Allows you to extract and display selected definitions. Note that each of the item names can be combined to provide shorter command lines such as the following:

```
$ RMU/EXTRACT/NOLOG/ITEMS=(ALL,NODATABASE) MF_PERSONNEL
```

If you omit the Items qualifier from the command line or specify it without any options, the action defaults to Items=All.

The following options can be specified with the Items qualifier:

- ♦ All
  Indicates that all database items are to be extracted. This is the default and includes all items except Alter_Database, Revoke_entry, Import, Load, Protections, Security, Unload, Verify, Volume, and Workload options. You can use either All or Noall in combination with other items to select specific output.
  In the following example, the Items=All option causes all the definitions except for Triggers to be extracted and displayed:

  ```
  $ RMU/EXTRACT/ITEMS=(ALL,NOTRIGGERS) MF_PERSONNEL
  ```

  The following example displays domain and table definitions. Note that the Noall option could have been omitted:

  ```
  $ RMU/EXTRACT/ITEMS=(NOALL, DOMAIN, TABLE) MF_PERSONNEL
  ```
- ♦ Alter_Database (or Change_Database)
  Displays the physical database after−image journal object definition.

- Catalog
  Displays all contents of the catalog created for an SQL multischema database. This item is ignored if the interface is RDO.
- Collating_Sequences
  Displays all the collating sequences defined for the database that you select. Note that Oracle Rdb does not save the name of the source OpenVMS National Character Set (NCS) library and the name becomes the defined logical, NCS$LIBRARY, by default.
- Constraints
  Noconstraints
  Table and column constraints are included in the output of the Items=Table qualifier. If you specify Item=Noconstraints, constraint information is not extracted for any table. If you specify the Language=SQL qualifier, the default is to have Item=Constraints enabled when tables are extracted.
  To extract all constraints as an ALTER TABLE statement, use the Item=Constraint and Option=Defer_Constraints qualifiers. To force all constraints to be defined after tables are defined, use the Item=Tables and Option=Defer_Constraints qualifiers.
- Database
  Displays the database attributes and characteristics. This includes information such as the database root file name, the number of buffers, the number of users, the repository path name, and the characteristics for each storage area.
  If you specify RMU Extract with the Option=Nodictionary_References qualifier, the data dictionary path name is ignored.
- Domains (or Fields)
  Displays the domain definitions. If the domain was originally defined using the data dictionary path name, the output definition shows this. If the Option=Nodictionary_References qualifier is specified, the data dictionary path name is ignored and the column attributes are extracted from the system tables.
- Functions
  Displays external function definitions.
- Import
  Generates an RDO or SQL IMPORT script that defines every storage area and row cache. The Language qualifier determines whether Oracle RMU generates an RDO or SQL IMPORT script (If you specify the Language=SQL or the Language=ANSI_SQL qualifier, the same SQL IMPORT script is generated.)
  Because the RDO interface does not accept many of the database options added to recent versions of Oracle Rdb, Oracle Corporation recommends that you specify the Language=SQL qualifier (or accept the default).
  The Items=Import qualifier is useful when you want to re−create a database that is the same or similar to an existing database. Editing the file generated by Oracle RMU to change allocation parameters or add storage areas and so on is easier than writing your own IMPORT script from scratch.
  When Oracle RMU generates the IMPORT script, it uses an interchange file name of rmuextract_rbr in the script. Therefore, you must either edit the IMPORT script generated by Oracle RMU to specify the interchange file that you want to import, or assign the logical name RMUEXTRACT_RBR to your interchange file name. (An interchange file is created by an SQL or RDO EXPORT statement.) See Example 14 in the Examples section.
- Indexes (or Indices)
  Displays index definitions, including storage map information.

- Load
  Unload
  Generates a DCL command procedure containing an RMU Load or RMU Unload command for each table in the database. This item must be specified explicitly, and is not included by default when you use the Items=All qualifier.
  Oracle RMU generates the Fields qualifier for the Load and Unload scripts when you specify the Option=Full qualifier. If you do not specify the Option=Full qualifier, the scripts are generated without the Fields qualifier.
  If you specify the RMU Extract command with the Item=Unload qualifier, DCL commands are added to the script to create a file with type .COLUMNS. This file defines all the unloaded columns. The file name of the .COLUMNS file is derived from the name of the extracted table. You can reference the file by using the "@" operator within the Fields qualifier for the RMU Load and RMU Unload commands.
- Module
  Displays procedure and function definitions. This item is valid only when the Language specification is SQL; it is ignored if the Language specification is RDO or ANSI_SQL.
- Outlines
  Displays query outline definitions. This item is valid only when the Language specification is SQL; it is ignored if the Language specification is RDO or ANSI_SQL.
- Procedures
  Extracts external procedures.
- Protections
  Displays the protection access control list (ACL) definitions. If the protections are defined using SQL ANSI semantics, they cannot be represented in RDO. In this case, the diagnostic message warns you that the protections must be extracted using the Language=SQL qualifier. If you specify Language=ANSI_SQL, a diagnostic message warns you that the ACL−style protections cannot be extracted in ANSI format. You must explicitly specify the Protections option. It is not included by default when you use the Items=All qualifier.
- Revoke_Entry
  Extracts a SQL or RDO script that deletes the protections from all access control lists in the database: database, table, column, module, function, and procedure.
  The output script contains a series of SQL REVOKE ENTRY statements (or DELETE PROTECTION statements if the language selected is RDO) that remove the access control entry for the user and all objects.
- Schema
  Displays the schema definitions for an SQL multischema database. This option is ignored if the interface is RDO.
- Security
  Displays RMU Audit commands based on information in the database. This item must be specified explicitly, and is not included by default when you use the Items=All qualifier.
- Storage_Maps
  Displays storage map definitions, including the list (segmented string) storage map.
- Tables (or Relations)
  Displays table definitions in the same order in which they were created in the database.
  If the table was originally defined using the data dictionary path name, that path name is used for the definition.

If you specify the Option=Nodictionary_References qualifier, the data dictionary path name is ignored and the table attributes are extracted from the system tables.
If Item=Noconstraints is specified, constraint information is not extracted for any table.
The Items=Tables qualifier handles domains in the following ways:

- ◊ The output for this item reflects the original definitions. If a column is based on a domain of a different name, the BASED ON clause is used in RDO, and the domain name is referenced by SQL.
- ◊ Any columns that are based on fields in a system table are processed but generate warning messages.
- ◊ Certain domains created using RDO in a relation definition cannot be extracted for RDO because it is not possible to distinguish columns defined using a shorthand method as shown in the example that follows. In this case, the column FIELD_1 becomes or is defined as a domain.

```
DEFINE RELATION REL1.
    FIELD_1     DATATYPE IS TEXT SIZE 10.
END.
```

However, this type of definition in SQL causes special domains to be created with names starting with SQL$. In this case, the SQL domain is translated into the following data type:

```
CREATE TABLE TAB1
   (COLUMN_1   CHAR(10));
```

The output for this item also includes the table−level constraints that can be applied: PRIMARY KEY, FOREIGN KEY, NOT NULL, UNIQUE, and CHECK. In the case of the CHECK constraint, the expression might not be translated to or from RDO and SQL due to interface differences.

- ♦ Triggers
  Displays trigger definitions.
- ♦ Verify
  Causes the generation of an optimal DCL command procedure containing multiple RMU Verify commands. Using this command procedure is equivalent to performing a full verification (RMU Verify with the All qualifier) for the database. This command procedure can be broken down further into partial command scripts to perform partial verify operations. These partial command scripts can then be submitted to different batch queues to do a full verify operation in parallel, or they can be used to spread out a full verify operation over several days by verifying a piece of the database at a time.
  A ***partitioning algorithm*** is a procedure to determine what portions of the database should be verified in the same command script. For example, areas with interrelations should be verified with the same partial command script. A partitioning algorithm considers the following when creating a partial command script from the equivalent RMU Verify command with the All qualifier:
  1. Each storage area is assigned to a partition.
  2. For each table in the database, if the table is not partitioned, the table is put in the partial command script corresponding to that storage area; otherwise, if the table is partitioned across several storage areas, the partitions corresponding to all of the storage areas are merged into one partial

command script and the table is added to this partial command script.
3. For each index in the database, the process shown in step 2 is followed.
4. For an index on a table, the index and table are merged into one partial command script.

The scripts of partial RMU Verify commands are written in the form of a command procedure. Each partial command script is preceded by a label of the form STREAM_n: where *n* is an integer greater than 1. For example, to execute the command at label STREAM_3:, invoke the command procedure by using the following syntax:

```
$ @<command-procedure-name> STREAM_3
```

The resultant command procedure is set up to accept up to four parameters, P1, P2, P3, and P4, as shown in Table 5−1.

Table 5−1 Parameters for Generated Command File

Parameter	Option	Description
P1	Stream_n	Specifies the command stream to be executed. The variable *n* is the "number" of the RMU Verify command stream to be executed. If omitted, all command streams are executed.
P2	[No]Log	Specifies whether to use the Log qualifier in the RMU Verify command line. If omitted, the DCL verify switch value is used.
P3	Read_Only | Protected | Exclusive	Provides the RMU Verify Transaction_Type value. If omitted, Transaction_Type = Protected is used.
P4		Specifies the name of the output file for the RMU Verify Output qualifier. If omitted, Output = SYS$OUTPUT is used.

- Views

  Displays view definitions. If the database was defined using SQL, it is possible that the view cannot be represented in RDO. In this case, the diagnostic message warns that the view definition is being ignored, and the user should use LANGUAGE=SQL to extract the view. Note the following transformations the RMU Extract command makes when it cannot precisely replicate the SQL source code:

  - The RMU Extract command cannot precisely replicate derived table column names or correlation names for any select expression.

    The RMU Extract command generates new names for correlation names (C followed by a number) and derived table column names (F followed by a number). In some cases the derived table column names will be inherited from the nested table columns.

    For example, suppose you create a view, as follows:

    ```
    SQL> ATTACH 'FILENAME mf_personnel';
    SQL> CREATE VIEW DERIVED_1
    ```

```
cont> (F1) AS
cont> SELECT CAST(AVG(JOB_COUNT) AS INTEGER(2))
cont> FROM (SELECT EMPLOYEE_ID, COUNT (*)
cont>   FROM JOB_HISTORY
cont>   GROUP BY EMPLOYEE_ID) AS EMP_JOBS (EMPLOYEE_ID, JOB_COUNT);
SQL> COMMIT;
```

If you issue the following RMU Extract command, you receive the output shown:

```
$ rmu/extract/item=view/option=(match:DERIVED_1%,noheader,filename_only) -
mf_personnel
set verify;
set language ENGLISH;
set default date format 'SQL92';
set quoting rules 'SQL92';
set date format DATE 001, TIME 001;
attach 'filename MF_PERSONNEL';
create view DERIVED_1
    (F1) as
    (select
        CAST(avg(C2.F2) AS INTEGER(2))
    from
        (select C4.EMPLOYEE_ID, count(*)
            from JOB_HISTORY C4
            group by C4.EMPLOYEE_ID)
        as C2 (F1, F2));

commit work;
```

◊ The RMU Extract command cannot generate the original SQL source code for the user−supplied names of AS clauses. This is particularly apparent when the renamed select expression is referenced in an ORDER BY clause. In such a case, the RMU Extract command generates correlation names in the form RMU$EXT_n where *n* is a number.
For example, suppose you create a view, as follows:

```
SQL> SET QUOTING RULES 'SQL92';
SQL> CREATE DATA FILE xyz;
SQL> CREATE TABLE DOCUMENT
cont> (REPORT CHAR(10));
SQL> CREATE TABLE REPORTING
cont> (NAME CHAR(5));
SQL> CREATE TABLE "TABLES"
cont> (CODTAB CHAR(5));
SQL> CREATE VIEW VIEW_TEST
cont> (CREDIT,
cont> CODTAB,
cont> CODMON) AS
cont> SELECT
cont> C1.NAME,
cont> C2.CODTAB,
cont> (SELECT C7.REPORT FROM DOCUMENT C7) AS COM
cont> FROM REPORTING C1, "TABLES" C2
cont> ORDER BY C1.NAME ASC, C2.CODTAB ASC, COM ASC;
SQL>
```

If you issue the following RMU Extract command, you receive the output

shown:

```
$ RMU/EXTRACT/ITEM=VIEW XYZ.RDB

   .
   .
   .
create view VIEW_TEST
    (CREDIT,
     CODTAB,
     CODMON) as
    select
        C1.NAME,
        C2.CODTAB,
        (select DOCUMENT.REPORT from DOCUMENT) AS RMU$EXT_1
    from REPORTING C1, "TABLES" C2
    order by C1."NAME" asc, C2.CODTAB asc, RMU$EXT_1 asc;
```

- Volume
  Displays cardinality information in a PDL−formatted file for use by Oracle Expert for Rdb. This item must be specified explicitly, and is not included by default when the Items=All qualifier is used.
- Workload
  Generates a DCL command language script. The script is used with the RMU Insert Optimizer_Statistics command to extract the work load and statistics stored in the RDB$WORKLOAD table. The unloaded information can be applied after a new database is created using the SQL EXPORT and IMPORT statements, or it can be applied to a similar database for use by the RMU Collect Optimizer_Statistics/Statistic=Workload command.
  This item must be specified explicitly, and is not included by default when the Items=All qualifier is used. The default is Noworkload.
  You can modify the output of the Item=Workload qualifier by specifying the following keywords with the Option qualifier:
  - ◊ Audit_Comment
    Each RMU Insert Optimizer_Statistics statement is preceded by the created and altered date for the workload entry. The default is Noaudit_comment.
  - ◊ Filename_Only
    The database file specification output for the RMU Insert Optimizer_Statistics statement is abbreviated to just the filename.
  - ◊ Match
    A subset of the workload entries based on the wildcard file name is selected.

Language=lang−name

Allows you to select one of the following interfaces:

- SQL
  When you specify the Language=SQL qualifier, Oracle RMU generates the Oracle Rdb SQL dialect. The Oracle Rdb SQL dialect is a superset of SQL92 Entry level, with language elements from Intermediate and Full SQL92 levels. It also contains language elements from SQL:1999 and extensions specific to Oracle Rdb.
- ANSI_SQL
  When you specify the Language=ANSI_SQL qualifier and specify the Option=Normal qualifier, Oracle RMU tries to generate ANSI SQL statements that conform to the ANSI X3.135−1989 SQL standard.

When you specify the Language=ANSI_SQL qualifier and the Option=Full qualifier, Oracle RMU tries to generate SQL statements that conform to the current ANSI and ISO SQL Database Language standards. Please refer to the Oracle Rdb SQL Reference Manual for details on the conforming SQL language standards. Regardless of the Option parameter you specify, any Oracle Rdb specific features (such as DATATRIEVE support clauses and storage maps) are omitted.

- RDO
  When you specify the RDO language option, Oracle RMU generates RDO statements.

The default is Language=SQL.

The Language qualifier has no effect on the output generated by the Items=Load, Items=Unload, and Items=Verify qualifiers. This is because these qualifiers generate scripts that contain Oracle RMU commands only.

Log[=log−file]

Nolog

Enable or disables log output during execution of the RMU Extract command. The log includes the current version number of Oracle Rdb, and the values of the parameter and qualifiers. The default is Nolog. The default file extension is .log. If you specify Log without specifying a file name, output is sent to SYS$OUTPUT.

Options=options−list

This qualifier is used to change the output of the RMU Extract command. This qualifier is not applied to output created by the Items=Unload, Items=Load, Items=Security, or the Items=Verify qualifier.

The following options can be specified with the Options qualifier:

- Audit_Comment
  Noaudit_Comment
  Annotates the extracted objects with the creation and last altered timestamps as well as the username of the creator. The date and time values are displayed using the current settings of SYS$LANGUAGE and LIB$DT_FORMAT. Noaudit_Comment is the default.
- Cdd_Constraints
  Nocdd_Constraints
  Specifies that tables extracted by pathname include all constraints. The Option=Nocdd_Constraints qualifier is equivalent to the Option=Defer_Constraints qualifier for tables with a pathname. This option is ignored if Item=Noconstraints is specified.
  When you specify the Cdd_Constraints option and the Dictionary_References option, the RMU Extract command does not generate ALTER TABLE statements to add constraints, but instead assumes they will be inherited from the data dictionary.
  When you use the Nocdd_Constraints option and the Dictionary_References option, the RMU Extract command generates ALTER TABLE statements to add FOREIGN KEY and CHECK constraints after all base tables have been created.

- Cdd_References
  This option is an alias for Dictionary_References.
- Column_Volume
  Directs the RMU Extract command to output the table, column, and column segmented string cardinalities based on sorted indexes. Note that this qualifier must be used in combination with the Items=Volume qualifier. If the Items=Volume qualifier is omitted, cardinalities are not displayed.
  RMU Extract generates data of the following type:

```
Volume for schema MF_PERSONNEL
    Default volatility is 5;
    Table WORK_STATUS all is 3;
    Table EMPLOYEES all is 100;
        Column EMPLOYEE_ID all is 100;
        Column LAST_NAME all is 83;
   .
   .
   .
    Table RESUMES all is 3;
        List RESUME
            Cardinality IS 3
                Number of segments is 3
                Average length of segments is 24;
```

- Debug
  Dumps the internal representation for SQL clauses such as VALID IF, COMPUTED BY, MISSING_VALUE, DEFAULT_VALUE, CONSTRAINTS, SQL DEFAULT, TRIGGERS, VIEWS, and STORAGE MAPS during processing. The keyword Debug cannot be specified with the keywords Normal or Full in the same Options qualifier list.
- Defer_Constraints
  Nodefer_Constraints
  Forces all constraints to be defined (using an ALTER TABLE statement) after all tables have been extracted. This option is ignored if Item=Noconstraints is specified. If Option=Nodefer_Constraints is specified, all constraints are generated with the CREATE TABLE statement. If neither Option=Defer_Constraints nor Option=Nodefer_Constraints is specified, the default behavior is to generate NOT NULL, UNIQUE, and PRIMARY KEY constraints with the CREATE TABLE statement, and generate CHECK and FOREIGN KEY constraints in a subsequent ALTER TABLE statement.
- Dictionary_References
  Nodictionary_References
  Directs the RMU Extract command to output definitions for domains (fields) and tables (relations) that reference data dictionary path names rather than using the information contained in the Oracle Rdb system tables. In addition to the database statements, this option also displays the data dictionary path name stored in the database. Refer to Example 8 in the Examples section for an example of using this option.
  If neither the Option=Dictionary_References qualifier nor the Option=Nodictionary_References qualifier is specified, then Oracle RMU examines the RDB$RELATIONS and RDB$FIELDS system tables to determine whether or not any domains or tables refer to the data dictionary. If references are made to the data dictionary, then the Option=Dictionary_References qualifier is the default. Otherwise, it is assumed that the data dictionary is not used, and the default is the

Option=Nodictionary_References qualifier.
The Nodictionary_References keyword causes all references to the data dictionary to be omitted from the output. This is desirable if the database definition is to be used on a system without the data dictionary or in a testing environment.
If the Items=Database and Option=Nodictionary_References qualifiers are selected, the data dictionary path name stored in the system table is ignored. For SQL, the no PATHNAME clause is generated, and for RDO, the clause DICTIONARY IS NOT USED is generated.
If the Items qualifier specifies Domain or Table, and the Option qualifier specifies Nodictionary_References, the output definition includes all attributes stored in the system tables.

- Disable_Objects
  Nodisable_Objects
  Requests that all disabled objects be written to the RMU Extract output file as disabled (see the description for the Omit_Disabled option). Disable_Objects is the default.
  The Nodisable_Objects option displays the objects but omits the disabling syntax.
- Domains
  Nodomains
  The Nodomains option is used to eliminate the domain name from within metadata objects. The domain name is replaced by the underlying data type. This option is designed for use with tools that do not recognize this SQL92 SQL language feature.
  Effect on /Language=SQL output:
  - The default is Option=Domains.
    A SQL script generated when Option=Nodomains was specified does not include the domain name in the CREATE TABLE column definition, CREATE FUNCTION or CREATE PROCEDURE parameter definitions, or any value expression which uses the CAST function to convert an expression to a domain data type (such as the CREATE VIEW and CREATE TRIGGER statements).
    The output generated by RMU Extract for functions and procedures in the CREATE MODULE statement is not affected by the Option=Nodomains option because it is based on the original source SQL for the routine body which is not edited by the RMU Extract command.

  Effect on /Language=ANSI_SQL output:
  - The default is Option=Nodomains when the Option=Normal qualifier is specified or is the default. The RMU Extract command does not generate a list of domain definitions even if the Items=Domains or Items=All qualifier is used. If you want the generated script to include a list of domain definitions, use the Options=Domains qualifier:

    ```
    $RMU/EXTRACT/LANGUAGE=ANSI_SQL/OPTION=DOMAINS databasename
    ```

    Use the Option=Full qualifier to have the use of domains included in the syntax generated for SQL92.
- Filename_Only
  Causes all file specifications extracted from the database to be truncated to only the file name. The use of this qualifier allows for easier relocation of the new database when you execute the created procedure.

- Full
  Nofull
  Specifies that if metadata that cannot be translated from the language that defined the database to the equivalent construct in the language specified with the Language qualifier (for example, DEFAULT for SQL and the language selected was RDO) then the metadata is displayed in comments, or Oracle RMU attempts to create a translation that most closely approximates the original construct.
  Nofull is identical to the Normal option.
- Group_Table
  Nogroup_Table
  Specifies that indexes and storage maps are to be extracted and grouped by table. The table is extracted first, than any PLACEMENT VIA index, then any storage map, and finally all other indexes.
  When the Group_Table qualifier is combined with the Option=Match qualifier, you can select a table and its related storage map and indexes.
  The default behavior is Nogroup_Table, which means that items are extracted and grouped by type.
- Header
  Noheader
  Specifies that the script header (and section headers) are included in the extract. This is the default. Noheader suppresses the header and because of the included date may allow easier comparison with other database extractions using the OpenVMS DIFFERENCES command.
- Limit_Volume=nn
  Nolimit_Volume
  Specifies the maximum amount of data to be scanned for segmented fields. The RMU Extract command stops scanning when the limit *nn* is reached. The number of segments and average length of segments are calculated from the data that was scanned. Limit_Volume=1000 is the default.
  Nolimit_Volume specifies that a full scan for segmented strings should be done.
- Match:match−string
  The Match option allows selection of wildcard object names from the database. The match string can contain the standard SQL wildcard characters: the percent sign (%) to match any number of characters, and the underscore (_) to match a single character. In addition, the backslash (\) can be used to prefix these wildcards to prevent them from being used in matching. If you are matching a literal backslash, use the backslash twice, as shown in the following example:

```
Option=Match:"A1\\A2%"
```

  The match string defaults to the percent sign (%) so that all objects are selected. To select those objects that start with JOB, use the qualifier Option=Match:"JOB%". From the mf_personnel database, this command displays the definitions for the domains JOB_CODE_DOM and JOB_TITLE_DOM, the tables JOBS and JOB_HISTORY, the index JOB_HISTORY_HASH, and the storage maps JOBS_MAP and JOB_HISTORY_MAP.
  The match string can be quoted as shown if the string contains spaces or other punctuation characters used by DCL or other command language interfaces. Most object names are space filled; therefore, follow the match string with the percent sign (%) to match all trailing spaces.

The Match option can be used in conjunction with the Item qualifier to extract specific tables, indexes, and so on, based on their name and type.
If Group_Table is specified, the match name is assumed to match a table name; indexes for that table will be extracted when the Items=Index qualifier is specified.

- Multischema
  Nomultischema
  Displays the SQL multischema names of database objects. It is ignored by the Relational Database Operator (RDO).
  The Nomultischema option displays only the SQL single−schema names of database objects.
- Normal
  Includes only the specific source language code used to define the database. This is the default.
  In addition, this option propagates RDO VALID IF clauses as column CHECK constraints with the attribute NOT DEFERRABLE when the Language specification is SQL or ANSI_SQL. When an RDO VALID IF clause is converted, Oracle RMU generates error messages similar to the following in your log file:

```
%RMU−W−UNSVALIDIF, VALID IF clause not supported in SQL − ignored
 for DEGREE.
%RMU−I−COLVALIDIF, changed VALID IF clause on domain DEGREE to
 column check constraint for DEGREES.DEGREE
```

  The first message is a warning that the VALID IF clause could not be added to the domain definition because the VALID IF clause is not supported by SQL. The second message is an informational message that tells you the VALID IF clause was changed to a column check constraint.
- Omit_Disabled
  Noomit_Disabled
  Causes all disabled objects to be omitted from the output of the RMU Extract command. This includes indexes that have MAINTENANCE IS DISABLED, USERS with ACCOUNT LOCK, and disabled triggers and constraints.
  The Noomit_Disabled option causes all disabled objects to be included in the output from the RMU Extract command. Noomit_Disabled is the default.
- Order_By_Name
  Noorder_By_Name
  Order_by_Name displays the storage area, cache, and journal names for the items Database, Alter_Database (also known as Change_Database), and Import in alphabetic order by the ASCII collating sequence.
  Noorder_By_Name displays the storage area, cache, and journal names for the items Database, Alter_Database, and Import in approximate definition order. The default ordering is approximate because a DROP STORAGE AREA, DROP CACHE, or DROP JOURNAL statement frees a slot that can be reused, thus changing the order. Noorder_By_Name is the default.
  You can use the logical name RDMS$BIND_SORT_WORKFILES to allocate work files, if needed.

Note

If the identifier character set for the database is not MCS or ASCII,

then this option is ignored. Characters from other character sets do not sort appropriately under the ASCII collating sequence.

- Volume_Scan
  Directs the RMU Extract command to perform queries to calculate the cardinality of each table, if both the Items=Volume and Options=Volume_Scan qualifiers are specified. The default is Options=Novolume_Scan, in which case the approximate cardinalities are read from the RDB$RELATIONS system table. The Options=Volume_Scan option is ignored if the Items=Volume qualifier is not selected.
- Width=n
  Specifies the width of the output files. You can select values from 60 to 512 characters. The default of 80 characters is appropriate for most applications.

Output=[out−file]

Nooutput

Names the file to which the RMU Extract command writes the data definition language (DDL) statements. The file extension defaults to .rdo, if you specify the Language=RDO qualifier; .sql, if you specify either the Language=SQL or the Language=ANSI_SQL qualifier. If you specify the Volume option only, the output file type defaults to .pdl. If you specify Load, Security, Verify, or Unload only, the output file type defaults to .com. The default is SYS$OUTPUT. If you disable the output by using the Nooutput qualifier, command scripts are not written to an output file. The Log output can be used to determine which features used by the database cannot be converted to SQL.

Table 5−2 shows the effects of the various combinations of the Language and Options qualifiers.

Table 5−2 Using Qualifiers to Determine Output Selection

Language	Option	Effect on Output
RDO	Normal	Generates RDO syntax.
	Full	Generates RDO syntax.
	Dictionary_ References	Outputs path name references to the repository.
	Nodictionary_ References	Converts path name references to the repository to RDO syntax.
	Multischema	Ignored by RDO.
SQL	Normal	Generates SQL syntax.
	Full	Tries to convert RDO specific features to SQL (for example, the VALID IF clause).
	Dictionary_ References	Outputs path name references to the data dictionary.
	Nodictionary_ References	Converts path name references to the data dictionary to SQL syntax.
	Multischema	Selects SQL multischema naming of objects.

ANSI_SQL	Normal	Generates ANSI/ISO syntax.
	Full	Generates ANSI/ISO SQL92 syntax supported by SQL.
	Dictionary_ References	Ignored for ANSI_SQL.
	Nodictionary_ References	Converts path name references to the data dictionary to SQL syntax. This is the default for ANSI_SQL.
	Multischema	Selects SQL multischema naming of objects.
Any	Audit_Comment	Adds a comment before each definition.
	Debug	Annotates output where possible.
	Domains	Replaces domain names for CAST expression, column and parameter definitions, and returns clauses with SQL data type.
	Filename_Only	Truncates all file specifications extracted from the database to only the file name.
	Volume_Scan	Forces a true count of Tables. Only valid for Items=Volume.

Transaction_Type[=(transaction_mode,options,...)]

Allows you to specify the transaction mode, isolation level, and wait behavior for transactions.

Use one of the following keywords to control the transaction mode:

- Automatic
  When Transaction_Type=Automatic is specified, the transaction type depends on the current database settings for snapshots (enabled, deferred, or disabled), transaction modes available to the process, and the standby status of the database. Automatic mode is the default.
- Read_Only
  Starts a READ ONLY transaction.
- Write
  Starts a READ WRITE transaction.

Use one of the following options with the keyword Isolation_Level=[level] to specify the transaction isolation level:

- Read_Committed
- Repeatable_Read
- Serializable. Serializable is the default setting.

Refer to the SET TRANSACTION statement in the Oracle Rdb SQL Reference Manual for a complete description of the transaction isolation levels.

Specify the wait setting by using one of the following keywords:

- Wait
  Waits indefinitely for a locked resource to become available. Wait is the default behavior.
- Wait=n
  The value you supply for *n* is the transaction lock timeout interval. When you supply

this value, Oracle Rdb waits *n* seconds before aborting the wait and the RMU Extract session. Specifying a wait timeout interval of zero is equivalent to specifying Nowait.

- Nowait
  Will not wait for a locked resource to become available.

Usage Notes

- To use the RMU Extract command for a database, you must have the RMU$UNLOAD privilege in the root file access control list (ACL) for the database or the OpenVMS SYSPRV or BYPASS privilege.
- For tutorial information on using output from the RMU Extract command to load or unload a database, refer to the *Oracle Rdb Guide to Database Design and Definition*.
- Included in the output from the RMU Extract command is the SQL SET DEFAULT DATE FORMAT statement. This SQL statement determines whether columns with the DATE data type or CURRENT_TIMESTAMP builtin function are interpreted as VMS or SQL92 format. The RMU Extract command always sets the default to SQL92. The SQL92 format DATE and CURRENT_TIMESTAMP contain only the YEAR to DAY fields. The VMS format DATE and CURRENT_TIMESTAMP contain YEAR to SECOND fields.
  If your database was defined with VMS format DATE and CURRENT_TIMESTAMP, the default SQL SET DEFAULT DATE FORMAT 'SQL92' in the Extract output causes errors to be returned when you attempt to execute that output. For example, when you define a trigger:

```
SQL> CREATE TRIGGER SALARY_HISTORY_CASCADE_UPDATE
cont>     AFTER UPDATE OF JOB_CODE ON JOB_HISTORY
cont>         (UPDATE SALARY_HISTORY SH
cont>             SET SALARY_START = CURRENT_TIMESTAMP
cont>             WHERE (SH.EMPLOYEE_ID = JOB_HISTORY.EMPLOYEE_ID)
cont>         ) for each row;
%SQL-F-UNSDATASS, Unsupported date/time assignment from <Source>
 to SALARY_START
```

  You can avoid these errors by editing the output from the RMU Extract command. Replace the SET DEFAULT DATE FORMAT 'SQL92' statement with SET DEFAULT DATE FORMAT 'VMS'. If the problem occurs in trigger definitions, you can use the CAST function instead. Specify CAST(CURRENT_TIMESTAMP AS DATE VMS) with each trigger definition that references CURRENT_TIMESTAMP. (You cannot use the CAST function within the DEFAULT clause of an SQL CREATE statement).
- The following list contains a description of what the RMU Extract command generates when it encounters certain RDO statements:
  - RDO and the data dictionary have the concept of validation clauses at the domain level. The ANSI/ISO SQL92 standard allows CHECK constraints defined on domains. While the actions of the ANSI/ISO CHECK constraint do differ from VALID IF in some respects, the RMU Extract command extracts the VALID IF clauses as domain CHECK constraints if you specify the Language=SQL and Option=Full qualifiers.
  - RDO multiline descriptions
    Because the RDO interface removes blank lines in multiline descriptions, the description saved in the metadata is not identical to that entered by the database definition. The RMU Extract command therefore cannot completely reconstruct the original description.
  - Some RDO trigger definitions
    RDO trigger definitions that contain a trigger action within a join of two or more tables

generates invalid SQL syntax. For example, the following RDO trigger definition includes a join with an embedded ERASE statement. When the RMU Extract command encounters this statement, Oracle RMU generates the invalid SQL trigger definition shown.

```
DEFINE TRIGGER EXAMPLE
    AFTER ERASE
    FOR C1 IN EMPLOYEES
    EXECUTE
        FOR C2 IN JOB_HISTORY
            CROSS C3 IN EMPLOYEES
            WITH (((C2.EMPLOYEE_ID = C3.EMPLOYEE_ID)
                 AND (C2.JOB_END MISSING))
                 AND (C3.EMPLOYEE_ID = C2.EMPLOYEE_ID))
            ERASE C2
        END_FOR
    FOR EACH RECORD.

CREATE TRIGGER EXAMPLE
    AFTER DELETE ON EMPLOYEES
        (DELETE FROM JOB_HISTORY C2, EMPLOYEES C3
            WHERE (((C2.EMPLOYEE_ID = C3.EMPLOYEE_ID)
                AND (C2.JOB_END IS NULL))
                AND (C3.EMPLOYEE_ID = C2.EMPLOYEE_ID))
        ) FOR EACH ROW;
```

Note that in Oracle Rdb Version 4.1 and higher, including a trigger action within a join of two or more tables is invalid RDO syntax. For more information on this RDO restriction, see the ERASE and MODIFY entries in RDO HELP.

- Oracle CDD/Repository Version 5.3 and higher support table and column constraint definition and maintenance through CDO. The RMU Extract command, by default, assumes all constraint maintenance is with SQL and so follows each CREATE TABLE with an ALTER TABLE FROM pathname to add the constraints. However, this is no longer necessary if you are using the later versions of Oracle CDD/Repository. To disable the output of the SQL ALTER TABLE statements which add constraints use the Option=Cdd_Constraint qualifier.
- If the Transaction_Type qualifier is omitted from the RMU Extract command line, a READ ONLY transaction is started against the database. This behavior is provided for backward compatibility with prior Oracle Rdb releases. If the Transaction_Type qualifier is specified without a transaction mode, the default value Automatic is used.
- If the database has snapshots disabled and the Transaction_Type qualifier was omitted, the transaction is restarted as READ WRITE ISOLATION LEVEL READ COMMITTED to reduce the number of rows locked by operations performed with the Option=Volume_Scan qualifier enabled.
- When Transaction_Type=Write is specified, the RMU Extract process does not attempt to write to the database tables.
- In previous versions, Oracle Rdb used derived column names based on position, for example, F1, F2. With release 7.0.6.4 and later, Oracle Rdb promotes the column names from the base table into the derived column name list. The result is a more readable representation of the view or trigger definition.
  In the following example the column name EMPLOYEE_ID is propagated through the derived table. In previous releases this would be named using a generic label F1.

```
create view SAMPLE_V
    (EMPLOYEE_ID,
     COUNTS) as
    select
```

```
      C1.EMPLOYEE_ID,
      C1.F2
from
  (select C2.EMPLOYEE_ID,
(select count(*) from SALARY_HISTORY C3
      where (C3.EMPLOYEE_ID = C2.EMPLOYEE_ID))
      from JOB_HISTORY C2) as C1 ( EMPLOYEE_ID, F2 )
  order by C1.F2 asc;
```

- The following list shows the equivalent SQL expressions matched by the RMU Extract process:
  - NULLIF (a, b) is eqivalent to

    ```
    CASE
      WHEN a = b THEN NULL
      ELSE a
    END
    ```
  - NVL (a, ..., b) or COALESCE (a, ..., b) is equivalent to

    ```
    CASE
      WHEN a IS NOT NULL THEN a
      ...
      ELSE b
    END
    ```
  - The simple CASE expression

    ```
    CASE a
      WHEN b THEN v1
      WHEN NULL THEN v2
      ...
      ELSE v3
    END
    ```

    is equivalent to

    ```
    CASE
      WHEN a = b THEN v1
      WHEN a IS NULL THEN v2
      ...
      ELSE v3
    END
    ```

    RMU Extract tries to decode the internal representation to as compact a SQL expression as possible.
- The RMU Extract process decodes case expressions into ABS (Absolute) functions:
  ABS(a) is equivalent to:

```
CASE
  WHEN a < 0 THEN −a
  ELSE a
END
```

In addition, similar forms of CASE expression are also converted to ABS:

```
CASE
  WHEN a <= 0 THEN −a
  ELSE a
END

CASE
  WHEN a > 0 THEN a
  ELSE −a
END

CASE
  WHEN a >= 0 THEN a
  ELSE −a
END
```

It is possible that the RMU Extract process will change existing CASE expressions into this more compact syntax, even if they were not originally coded as an ABS function call.

- If the Group_Table option is used and the Item qualifier omits one or more of the Table, Index, or Storage_Map keywords, only the included items are displayed. For example, to extract just the indexes for the EMPLOYEES table:

```
$ RMU/EXTRACT/ITEM=INDEX/OPTION=(GROUP_TABLE,MATCH=EMPLOYEES%)
```

To extract only the storage map and indexes for a table, use the following command:

```
$ RMU/EXTRACT/ITEM=(STORAGE_MAP,INDEX)/OPTION=(GROUP_TABLE, −
_$ MATCH=EMPLOYEES%)
```

- If the name of the LIST storage map is not known, it can be extracted using the following generic command:

```
$ RMU/EXTRACT/ITEM=STORAGE_MAP/OPTION=(GROUP_TABLE, −
_$ MATCH=RDB$SEGMENTED_STRING%)
```

Examples

Example 1

The following command extracts these database items: COLLATING_SEQUENCES, DOMAINS, TABLES, INDEXES, STORAGE_MAPS, VIEWS, and TRIGGERS.

The All option is the default. The All or Noall option can be used in conjunction with other items to select specific output. For example, the Items=(All,Nodatabase) qualifier selects all metadata items except the physical database characteristics.

```
$ RMU/EXTRACT/ITEM=(ALL, NODATABASE) MF_PERSONNEL
```

Example 2

The following command generates a DCL command procedure containing an RMU Load command for each table in the database:

```
$ RMU/EXTRACT/ITEMS=LOAD MF_PERSONNEL
```

Example 3

The following command displays the protection access control list (ACL) definitions in the mf_personnel.rdb database:

```
$ RMU/EXTRACT/ITEMS=PROTECTIONS MF_PERSONNEL.RDB
```

Example 4

The following command generates a DCL command procedure containing an RMU Unload command for each table in the database:

```
$ RMU/EXTRACT/ITEMS=UNLOAD MF_PERSONNEL.RDB
```

Example 5

The following example displays index definitions:

```
$ RMU/EXTRACT/ITEMS=INDEXES MF_PERSONNEL
```

Example 6

The following example displays domain and table definitions. Note that the Noall option could have been omitted.

```
$ RMU/EXTRACT/ITEMS=(NOALL,DOMAINS,TABLES) MF_PERSONNEL
```

Example 7

The following example displays definitions for domains (fields) and tables (relations) that reference data dictionary path names rather than using the information contained in the Oracle Rdb system tables. In addition to the database statements, it also references the data dictionary path name stored in the database, as shown in the following example:

```
$ RMU/EXTRACT/LANG=SQL/ITEM=ALL/OPTION=DIC/OUTPUT=CDD_MODEL.LOG/LOG= -
_$ CDD_EXTRACT.LOG CDD_SQL_DB
```

Example 8

The following example creates a command procedure containing a script of partial RMU Verify commands or verify command partitions for the mf_personnel database. This command procedure was created with the following RMU Extract command:

```
$ RMU/EXTRACT/ITEM=VERIFY MF_PERSONNEL
```

Example 9

The following command displays a query outline definition that was previously added to the mf_personnel database:

```
$ RMU/EXTRACT/ITEMS=(OUTLINES) MF_PERSONNEL
```

Example 10

The following command displays the after−image journal (.aij) file configuration for mf_personnel:

```
$ RMU/EXTRACT/ITEMS=(ALTER_DATABASE) MF_PERSONNEL
```

Example 11

The following command displays the function definitions in mf_personnel for functions previously created using SQL:

```
$ RMU/EXTRACT/ITEM=FUNCTION MF_PERSONNEL
```

Example 12

The following command displays the table and column cardinalities based on sorted indexes:

```
$ RMU/EXTRACT/OPTION=COLUMN_VOLUME/ITEM=VOLUME MF_PERSONNEL
```

Example 13

The following example:

- Executes an SQL EXPORT statement to create an interchange file.
- Executes an RMU Extract command with the Item=Import qualifier to generate an Import script. In addition, the Option=Filename_Only qualifier is specified to prevent full file specifications from appearing in the SQL IMPORT script. (If full file specifications are used, you cannot test the script without replacing the database that was exported.)
- Defines a logical to define the interchange file name used in the Import script file.
- Executes the Import script file.

```
SQL> -- Create interchange file, SAVED_PERS.RBR.
SQL> --
SQL> EXPORT DATABASE FILENAME MF_PERSONNEL.RDB INTO SAVED_PERS.RBR;
SQL> EXIT;
$ !
$ RMU/EXTRACT/ITEM=IMPORT/OPTION=FILENAME_ONLY/OUTPUT=IMPORT_PERS.SQL -
_$ MF_PERSONNEL
$ DEFINE/USER RMUEXTRACT_RBR SAVED_PERS.RBR
$ !
$ SQL
SQL> @IMPORT_PERS.SQL
SQL> set language ENGLISH;
SQL> set default date format 'SQL92';
SQL> set quoting rules 'SQL92';
SQL> set date format DATE 001, TIME 001;
SQL>
SQL> -- RMU/EXTRACT for Oracle Rdb V7.0-00     1-JUL-1996 15:34:38.63
SQL> --
SQL> --                          Physical Database Definition
SQL> --
SQL> ------------------------------------------------------------------
SQL> import database from rmuextract_rbr
cont>     filename 'MF_PERSONNEL'
```

```
   .
   .
   .
```

Example 14

The following example shows an extract from the generated script when the SYS$LANGUAGE and LIB$DT_FORMAT symbols are defined. The language and format will default to ENGLISH and the standard OpenVMS format if these logical names are not defined.

```
$ DEFINE LIB$DT_FORMAT LIB$DATE_FORMAT_002,LIB$TIME_FORMAT_001
$ DEFINE SYS$LANGUAGE french
$ RMU/EXTRACT/OUT=SYS$OUTPUT/ITEM=DOMAIN MF_PERSONNEL/OPT=AUDIT_COMMENT
  .
  .
  .
-- Created on  8 janvier 2000 13:01:31.20
-- Never altered
-- Created by RDB_EXECUTE
--
SQL> CREATE DOMAIN ADDRESS_DATA_1
cont> CHAR (25)
cont> comment on domain ADDRESS_DATA_1 is
cont>   ' Street name';
  .
  .
  .
```

Example 15

If a database has snapshots set to ENABLED DEFERRED, it may be preferable to start a read/write transaction. In this environment, using the Transaction_type=(Read_only) qualifier causes a switch to a temporary snapshots ENABLED IMMEDIATE state. This transition forces the READ ONLY transaction to wait while all READ WRITE transactions complete, and then all new READ WRITE transactions performing updates will start writing rows to the snapshot files for use by possible read−only transactions. To avoid this problem use an RMU Extract command specifying a READ WRITE ISOLATION LEVEL READ COMMITTED transaction.

```
$ RMU/EXTRACT/ITEM=TABLE/OUT=TABLES.SQL−
    /TRANSACTION_TYPE=(WRITE,ISOLATION=READ)−
    SAMPLE.RDB
```

Example 16

This example specifies the options which were the default transaction style in prior releases.

```
$ RMU/EXTRACT/ITEM=TABLE/OUT=TABLES.SQL−
    /TRANSACTION_TYPE=(READ_ONLY)−
    SAMPLE.RDB
```

Example 17

If the database currently has snapshots deferred, it may be more efficient to start a read−write transaction with isolation level read committed. This allows the transaction to start immediately (a read−only transaction may stall), and the selected isolation level keeps row locking to a minimum. This could be explicitly stated by

using the following command:

```
$ RMU/EXTRACT-
    /TRANSACTION_TYPE=(WRITE,ISOLATION=READ_COMMITTED)-
    SAMPLE.RDB
```

Using a transaction type of automatic adapts to different database settings:

```
$ RMU/EXTRACT-
    /TRANSACTION_TYPE=(AUTOMATIC)-
    SAMPLE.RDB
```

Example 18

This example shows the use of the Item=Workload qualifier to create a DCL command language script for OpenVMS systems.

```
$ RMU/EXTRACT/ITEM=WORKLOAD -
    SCRATCH/LOG/OPTION=(FILENAME,AUDIT)
$! RMU/EXTRACT for Oracle Rdb X7.0-00            7-SEP-2000 22:00:42.72
$!
$!                               WORKLOAD Procedure
$!
$!------------------------------------------------------------------------
$ SET VERIFY
$ SET NOON
$
$! Created on  7-SEP-2000 10:12:26.36
$! Last collected on  7-SEP-2000 22:00:34.47
$!
$ RMU/INSERT OPTIMIZER_STATISTICS -
  SCRATCH -
  /TABLE=(CUSTOMERS) -
  /COLUMN_GROUP=(CUSTOMER_NAME) -
  /DUPLICITY_FACTOR=(4.0000000) -
  /NULL_FACTOR=(0.0000000) /LOG
$
$! Created on  7-SEP-2000 10:12:26.36
$! Last collected on  7-SEP-2000 22:00:34.58
$!
$ RMU/INSERT OPTIMIZER_STATISTICS -
  SCRATCH -
  /TABLE=(RDB$FIELDS) -
  /COLUMN_GROUP=(RDB$FIELD_NAME) -
  /DUPLICITY_FACTOR=(1.7794118) -
  /NULL_FACTOR=(0.0000000) /LOG
$
   .
   .
   .
$ SET NOVERIFY
$ EXIT
```

Example 19

The following example shows the use of the Match option to select a subset of the workload entries based on the wildcard file name.

```
$ RMU/EXTRACT/ITEM=WORKLOAD -
     SCRATCH/LOG/OPTION=(FILENAME,AUDIT,MATCH:RDB$FIELDS%)
$! RMU/EXTRACT for Oracle Rdb X7.0-00                    8-SEP-2000 10:53
$!
$!                              WORKLOAD Procedure
$!
$!-----------------------------------------------------------------------
$ SET VERIFY
$ SET NOON
$
$! Created on  7-SEP-2000 15:18:02.30
$! Last collected on  7-SEP-2000 18:25:04.27
$!
$ RMU/INSERT OPTIMIZER_STATISTICS -
  SCRATCH -
  /TABLE=(RDB$FIELDS) -
  /COLUMN_GROUP=(RDB$FIELD_NAME) -
  /DUPLICITY_FACTOR=(1.0000000) -
  /NULL_FACTOR=(0.0000000) /LOG
$ SET NOVERIFY
$ EXIT
```

Example 20

The following example shows the use of Item options Defer_Constraints, Constraints, and Match to extract a table and its constraints.

```
$ RMU/EXTRACT/ITEM=(TABLE,CONSTRAINT)-
_$ /OPTION=(FILENAME_ONLY,NOHEADER,-
_$          DEFER_CONSTRAINT,MATCH:EMPLOYEES%) -
_$ MF_PERSONNEL
set verify;
set language ENGLISH;
set default date format 'SQL92';
set quoting rules 'SQL92';
set date format DATE 001, TIME 001;
attach 'filename MF_PERSONNEL';
create table EMPLOYEES (
    EMPLOYEE_ID ID_DOM,
    LAST_NAME LAST_NAME_DOM,
    FIRST_NAME FIRST_NAME_DOM,
    MIDDLE_INITIAL MIDDLE_INITIAL_DOM,
    ADDRESS_DATA_1 ADDRESS_DATA_1_DOM,
    ADDRESS_DATA_2 ADDRESS_DATA_2_DOM,
    CITY CITY_DOM,
    STATE STATE_DOM,
    POSTAL_CODE POSTAL_CODE_DOM,
    SEX SEX_DOM,
    BIRTHDAY DATE_DOM,
    STATUS_CODE STATUS_CODE_DOM);
    comment on table EMPLOYEES is
      'personal information about each employee';

alter table EMPLOYEES
    add constraint EMP_SEX_VALUES
        check(EMPLOYEES.SEX in ('M', 'F', '?'))
        deferrable
    add constraint EMP_STATUS_CODE_VALUES
        check(EMPLOYEES.STATUS_CODE in ('0', '1', '2', 'N'))
        deferrable
    alter column EMPLOYEE_ID
```

```
        constraint EMPLOYEES_PRIMARY_EMPLOYEE_ID
            primary key
            deferrable;

commit work;
```

Example 21

The following example shows the use of the option Group_Table to extract a table and its indexes:

```
$ rmu/extract/item=(table,index)-
_$ /option=(group_table,match=employees%,-
_$          filename_only,noheader) db$:mf_personnel
set verify;
set language ENGLISH;
set default date format 'SQL92';
set quoting rules 'SQL92';
set date format DATE 001, TIME 001;
attach 'filename MF_PERSONNEL';
create table EMPLOYEES (
    EMPLOYEE_ID ID_DOM
        constraint EMPLOYEES_PRIMARY_EMPLOYEE_ID
            primary key
            deferrable,
    LAST_NAME LAST_NAME_DOM,
    FIRST_NAME FIRST_NAME_DOM,
    MIDDLE_INITIAL MIDDLE_INITIAL_DOM,
    ADDRESS_DATA_1 ADDRESS_DATA_1_DOM,
    ADDRESS_DATA_2 ADDRESS_DATA_2_DOM,
    CITY CITY_DOM,
    STATE STATE_DOM,
    POSTAL_CODE POSTAL_CODE_DOM,
    SEX SEX_DOM,
    BIRTHDAY DATE_DOM,
    STATUS_CODE STATUS_CODE_DOM);
    comment on table EMPLOYEES is
      'personal information about each employee';

    create unique index EMPLOYEES_HASH
        on EMPLOYEES (
        EMPLOYEE_ID)
        type is HASHED SCATTERED
        store
            using (EMPLOYEE_ID)
                in EMPIDS_LOW
                    with limit of ('00200')
                in EMPIDS_MID
                    with limit of ('00400')
                otherwise in EMPIDS_OVER;

    create unique index EMP_EMPLOYEE_ID
        on EMPLOYEES (
        EMPLOYEE_ID
            asc)
        type is SORTED
        node size 430
        disable compression;

    create index EMP_LAST_NAME
        on EMPLOYEES (
```

```
        LAST_NAME
            asc)
        type is SORTED;

commit work;

alter table EMPLOYEES
    add constraint EMP_SEX_VALUES
        check(EMPLOYEES.SEX in ('M', 'F', '?'))
        deferrable
    add constraint EMP_STATUS_CODE_VALUES
        check(EMPLOYEES.STATUS_CODE in ('0', '1', '2', 'N'))
        deferrable;

commit work;
```

Example 22

The following example shows the output when you use the Item=Revoke_Entry qualifier:

```
$ RMU/EXTRACT/ITEM=REVOKE_ENTRY ACCOUNTING_DB
...
--                              Protection Deletions
--
------------------------------------------------------------------------------

revoke entry
    on database alias RDB$DBHANDLE
    from [RDB,JAIN];

revoke entry
    on database alias RDB$DBHANDLE
    from [RDB,JONES];

revoke entry
    on database alias RDB$DBHANDLE
    from PUBLIC;

revoke entry
    on table ACCOUNT
    from [RDB,JONES];

revoke entry
    on table ACCOUNT
    from PUBLIC;

revoke entry
    on table ACCOUNT_BATCH_PROCESSING
    from [RDB,JONES];

revoke entry
    on table ACCOUNT_BATCH_PROCESSING
    from PUBLIC;
revoke entry
    on table BILL
    from [RDB,JONES];

revoke entry
    on table BILL
    from PUBLIC;
```

```
...
```

Example 23

The following example shows sample output for the WORK_STATUS table of MF_PERSONNEL. The uppercase DCL commands are generated by RMU Extract.

```
$ RMU/EXTRACT/ITEM=UNLOAD-
_$ /OPTION=(NOHEADER,FULL,MATCH:WORK_STATUS%) sql$database
$ CREATE WORK_STATUS.COLUMNS
! Columns list for table WORK_STATUS
! in DISK1:[DATABASES]MF_PERSONNEL.RDB
! Created by RMU Extract for Oracle Rdb V7.0-00 on  1-MAR-2001 20:50:25.33
STATUS_CODE
STATUS_NAME
STATUS_TYPE
$ RMU/UNLOAD -
        DISK1:[DATABASES]MF_PERSONNEL.RDB -
        /FIELDS="@WORK_STATUS.COLUMNS" -
        WORK_STATUS -
        WORK_STATUS.UNL
$
$ EXIT

$ RMU/EXTRACT/ITEM=LOAD-
_$ /OPTION=(NOHEADER,FULL,MATCH:WORK_STATUS%) sql$database
$ RMU/LOAD -
        /TRANSACTION_TYPE=EXCLUSIVE -
        /FIELDS="@WORK_STATUS.COLUMNS" -
        DISK1:[DATABASES]MF_PERSONNEL.RDB -
        WORK_STATUS -
        WORK_STATUS.UNL
$
$ EXIT
```

Example 24

The following example shows how to extract all constraints as an ALTER TABLE statement.

```
$ rmu/extract/item=(notab,constr) db$:sql_personnel/opt=(nohead,mat=empl%,defer)
set verify;
set language ENGLISH;
set default date format 'SQL92';
set quoting rules 'SQL92';
set date format DATE 001, TIME 001;
attach 'filename MF_PERSONNEL';
alter table EMPLOYEES
    add constraint EMP_SEX_VALUES
        check(EMPLOYEES.SEX in ('M', 'F', '?'))
        deferrable
    add constraint EMP_STATUS_CODE_VALUES
        check(EMPLOYEES.STATUS_CODE in ('0', '1', '2', 'N'))
        deferrable
    alter column EMPLOYEE_ID
        constraint EMPLOYEES_PRIMARY_EMPLOYEE_ID
            primary key
            deferrable;

commit work;
```

Chapter 6
Enhancements Provided in Previous Releases

6.1 Enhancements Provided in Oracle Rdb Release 7.0.7.2

6.1.1 Rdb Optional Site−Specific Startup Procedure

The Oracle Rdb startup procedure RMONSTART(xx).COM now supports an optional site−specific startup procedure to be executed after the Rdb Monitor (RDMMON) process has been started. If the file SYS$STARTUP:RDB$SYSTARTUP(xx).COM (where xx indicates the version number for multi−version Rdb kits) is found, it is executed as a DCL command procedure by the RMONSTART(xx).COM procedure.

SYS$STARTUP:RDB$SYSTARTUP(xx).COM is intended to contain site−specific tasks to be executed after the Rdb monitor procedure has completed. Such tasks might include opening databases or starting layered products that depend on the Rdb monitor process having been started.

If a site wishes to use this capability, the RDB$SYSTARTUP(xx).COM procedure must be created in SYS$STARTUP (either in the common SYS$COMMON:[SYS$STARTUP] directory or a node−specific SYS$SPECIFIC:[SYS$STARTUP] directory). The Rdb installation procedure does not provide or replace this file.

6.1.2 Oracle Rdb SGA API

Oracle Rdb maintains an extensive set of online performance statistics that provide valuable dynamic information regarding the status of an active database. The system global area (SGA) application programming interface (API) described in this document provides a way to retrieve these database performance statistics.

The SGA API automates retrieving database statistics available only through the RMU Show Statistics command. The SGA API provides the only way to retrieve statistics for Oracle Rdb databases from an application. Using the SGA API provides fast access to the data without effecting the execution of the server.

Previously, the Oracle Rdb SGA API was available as a separate software option to be downloaded, installed and maintained independently of the Oracle Rdb kit. Each time a new version of Oracle Rdb was installed, the SGA API would have to be updated. If the SGA API was not updated, it would, in many cases, fail to work correctly.

This problem has been partly resolved. Most of the contents of the SGA API separate software option are now automatically provided in the RDM$DEMO directory during the Oracle Rdb kit installation procedure. Please refer to the SGA API documentation available in RDM$DEMO in various formats as SGAAPI.PS, SGAAPI.HTML and SGAAPI.TXT for additional information.

Existing Users of the SGAAPI May Have to Modify Procedures

Existing users of the Oracle Rdb SGA API should refer to the documentation as there will be some minor changes required. In particular, the KUSRMUSHRxx.EXE sharable image is now provided in SYS$LIBRARY and the KUSRMUSHRxx.OPT linker options file has been updated to reference this sharable image in its new location.

6.1.3 CHRONO_FLAG Replaces Older CRONO_FLAG Keyword

The SET FLAGS statement and the RDMS$SET_FLAGS logical now accept the keyword CHRONO_FLAG as an alternate spelling of the existing keyword CRONO_FLAG. The latter keyword is deprecated and will be removed from a future release of Rdb V7.1.

The new keyword will also be displayed by the SHOW FLAGS statement.

6.2 Enhancements Provided in Oracle Rdb Release 7.0.7.1

6.2.1 RDM$BIND_SNAP_QUIET_POINT Logical No Longer Used

Bug 2656534

If the logical RDM$BIND_SNAP_QUIET_POINT was defined to be "0" on a system that was used for the standby database in a Hot Standby configuration, it was not possible to start database replication. Attempts to start replication would fail with:

```
RDMS−F−HOTSNAPQUIET, quiet points must be enabled
for snapshot transactions during hot standby replication
```

However, defining the logical to "1" can cause processes with long running READ ONLY transactions to prevent database backups from proceeding.

This logical was introduced in Oracle Rdb Release 6.0 to allow database administrators to override the 6.0 requirement that READ ONLY transactions hold the quiet−point lock. Defining the logical to "1" (the default) would provide better performance when the Fast Commit feature was enabled and processes frequently switched between READ ONLY and READ WRITE transactions. Since that time, improvements have been made to the quiet−point lock algorithms that make this logical no longer necessary. Since releases 7.0.6.3 and 7.1.0.1, READ ONLY transactions would continue to hold the quiet−point lock until a backup process requested the lock. When the lock was requested, READ ONLY processes would release the quiet point lock as soon as the currently executing database request finished, if the RDM$BIND_SNAP_QUIET_POINT logical was defined as "0". This made it no longer necessary to have the logical defined as "1".

Since the quiet−point lock behavior now behaves optimally, even with the Fast Commit feature enabled, the RDM$BIND_SNAP_QUIET_POINT logical is no longer needed and thus has been removed. Oracle Rdb will now behave as if the logical is always defined to be "0".

This problem has been corrected in Oracle Rdb Release 7.0.7.1.

6.2.2 Determining Which Oracle Rdb Options Are Installed

When installing Oracle Rdb Server on OpenVMS you can choose from five components to install:

1. Oracle Rdb
2. Programmer for Rdb (Rdb Compilers)
3. Hot Standby
4. Power Utilities
5. Common Components

Starting with Rdb 7.0, you can determine what Rdb options were selected during the installation of Rdb by running the program SYS$SYSTEM:RDBINS<RdbVersionVariant>.EXE. For example:

```
$ RUN SYS$SYSTEM:RDBINS70
Installed: Oracle Rdb,Rdb Compilers,Hot Standby,Power Utilities
```

Previously, however, the output of the RDBINS program could not easily be redirected. Attempts to redefine SYS$OUTPUT would not allow the program output to be captured.

This problem has been resolved. The RDBINS program now allows redirection of the output to SYS$OUTPUT. The RDBINS program also creates a DCL symbol RDB$INSTALLED_SELECTIONS containing the same output string as is displayed to SYS$OUTPUT.

6.2.3 New Procedure RDB$IMAGE_VERSIONS.COM

The command procedure RDB$IMAGE_VERSIONS.COM is supplied in SYS$LIBRARY by the Rdb installation procedure. The RDB$IMAGE_VERSIONS command procedure can be used to display the image identification string and image link date/time from various Oracle Rdb or potentially related images in SYS$SYSTEM, SYS$LIBRARY and SYS$MESSAGE. This procedure can be used to determine exactly what images are installed on the system.

RDB$IMAGE_VERSIONS.COM accepts an optional parameter. If passed, this parameter specifies a specific file or wildcard to lookup and display information for. By default, filenames starting with RD*, SQL*, RM*, and COSI* and ending with .EXE are searched for and displayed.

The following example shows how to use the RDB$IMAGE_VERSIONS command procedure.

```
Decrdb RTA1:> @RDB$IMAGE_VERSIONS
SYS$SYSROOT:[SYSEXE]RDB$NATCONN71.EXE;1 SQL*NET V7.1-55   8-MAY-2002 15:56
SYS$COMMON:[SYSEXE]RDBINS.EXE;4         ORACLE RDB V7.0  14-NOV-2002 17:20
SYS$COMMON:[SYSEXE]RDBINS70.EXE;33      ORACLE RDB V7.0   7-MAR-2003 15:30
SYS$COMMON:[SYSEXE]RDBINS71.EXE;5       ORACLE RDB V7.1   9-APR-2003 10:58
SYS$COMMON:[SYSEXE]RDBPRE.EXE;5         V7.0-65          10-SEP-2002 16:02
SYS$COMMON:[SYSEXE]RDBPRE70.EXE;37      V7.0-7           28-FEB-2003 23:24
SYS$COMMON:[SYSEXE]RDBPRE71.EXE;5       V7.1-101          8-APR-2003 16:49
SYS$COMMON:[SYSEXE]RDBSERVER.EXE;9      RDB/RSV V7.0-65   5-SEP-2002 21:01
SYS$COMMON:[SYSEXE]RDBSERVER70.EXE;41   RDB/RSV V7.0-7   27-FEB-2003 17:29
SYS$COMMON:[SYSEXE]RDBSERVER71.EXE;5    RDB/RSVV7.1-101   7-APR-2003 17:43
SYS$COMMON:[SYSEXE]RDMABS.EXE;5         RDB V7.0-65      10-SEP-2002 16:01
    .
    .
    .
```

Chapter 7
Documentation Corrections

This chapter provides information not currently available in the Oracle Rdb documentation set.

7.1 Documentation Corrections

7.1.1 Database Server Process Priority Clarification

By default, the database servers (ABS, ALS, DBR, LCS, LRS, RCS) created by the Rdb monitor inherit their VMS process scheduling base priority from the Rdb monitor process. The default priority for the Rdb monitor process is 15.

Individual server priorities can be explicitly controlled via system−wide logical names as described in Table 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.

7.1.2 Waiting for Client Lock Message

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

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

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

These metadata locks consist of three longwords. The lock is displayed in text format first, followed by its hexadecimal representation. The text version masks out non−printable characters with a dot (.).

The leftmost value seen in the hexadecimal output contains the id of the object. The id is described below for tables and views, functions, procedures, and modules.

- For tables and views, the id represents the unique value found in the RDB$RELATION_ID column of the RDB$RELATIONS system relation for the given table.
- For routines, the id represents the unique value found in the RDB$ROUTINE_ID column of the RDB$ROUTINES system relation for the given routine.
- For modules, the id represents the unique value found in the RDB$MODULE_ID column of the RDB$MODULES system relation for the given module.

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

Table 7−2 Objects and Their Hexadecimal Type Value

Object	Hexadecimal Value
Tables or views	00000004
Routines	00000016
Modules	00000015

The last value in the hexadecimal output represents the lock type. The value 55 indicates this is a client lock.

The following example shows a "waiting for client lock" message from a Stall Messages screen:

```
Process.ID Since.....   Stall.reason.......................... Lock.ID.
46001105:2 10:40:46.38 − waiting for client '........' 0000001900000004000000055
```

To determine the name of the referenced object given the lock ID, the following queries can be used based on the object type:

```
SQL> select RDB$RELATION_NAME from RDB$RELATIONS where RDB$RELATION_ID = 25;
SQL> select RDB$MODULE_NAME from RDB$MODULES where RDB$MODULE_ID = 12;
SQL> select RDB$ROUTINE_NAME from RDB$ROUTINES where RDB$ROUTINE_ID = 7;
```

Because the full client lock output is long, it may require more space than is allotted for the Stall.reason column and therefore can be overwritten by the Lock.ID. column output.

For more detailed lock information, perform the following steps:

- Press the L option from the horizontal menu to display a menu of lock IDs.
- Select the desired lock ID.

7.1.3 Clarification of PREPARE Statement Behavior

Bug 2581863

According to the Oracle Rdb7 SQL Reference Manual, Volume 3 page 7−227, when using a statement−id parameter for PREPARE "if that parameter is an integer, then you must explicitly initialize that integer to zero before executing the PREPARE statement".

This description is not correct and should be replaced with this information:

1. If the statement−id is non−zero and does not match any prepared statement (the id was stale or contained a random value), then an error is raised:
   %SQL−F−BADPREPARE, Cannot use DESCRIBE or EXECUTE on a statement that is not prepared
2. If the statement−id is non−zero, or the statement name is one that has previously been used and matches an existing prepared statement, then that statement is automatically released prior to the prepare of the new statement. Please refer to the RELEASE statement for further details.
3. If the statement−id is zero or was automatically released, then a new statement−id is allocated and the statement prepared.

Please note that if you use statement−name instead of a statement−id−parameter then SQL will implicitly declare an id for use by the application. Therefore, the semantics described apply similarly when using the statement−name. See the RELEASE statement for details.

7.1.4 SQL EXPORT Does Not Save Some Database Attributes

Bug 2574640

The SQL EXPORT and IMPORT commands do not support several database attributes.

The following attributes are not saved by EXPORT for this release of Oracle Rdb.

- CHECKPOINT TIMED EVERY n SECONDS
- CHECKPOINT { ALL | UPDATED } ROWS TO { BACKING FILE | DATABASE }
- RMU/SET LOGMINER

This problem has been corrected in Oracle Rdb Release 7.1. The EXPORT protocol has been extended to support these and other database attributes. If you have databases with these attributes enabled then after the IMPORT completes, use a SQL script to re−establish the settings for the new database.

7.1.5 RDM$BIND_LOCK_TIMEOUT_INTERVAL Overrides the Database Parameter

Bug 2203700

When starting a transaction, there are three different values that are used to determine the lock timeout interval for that transaction. Those values are:

1. The value specified in the SET TRANSACTION statement
2. The value stored in the database as specified in CREATE or ALTER DATABASE
3. The value of the logical name RDM$BIND_LOCK_TIMEOUT_INTERVAL

The timeout interval for a transaction is the smaller of the value specified in the SET TRANSACTION statement and the value specified in CREATE DATABASE. However, if the logical name RDM$BIND_LOCK_TIMEOUT_INTERVAL is defined, the value of this logical name overrides the value specified in CREATE DATABASE.

The description of how these three values interact, found in several different parts of the Rdb documentation set, is incorrect and will be replaced by the description above.

The lock timeout value in the database can be dynamically modified from the Locking Dashboard in RMU/SHOW STATISTICS. The Per−Process Locking Dashboard can be used to dynamically override the logical name RDM$BIND_LOCK_TIMEOUT_INTERVAL for one or more processes.

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

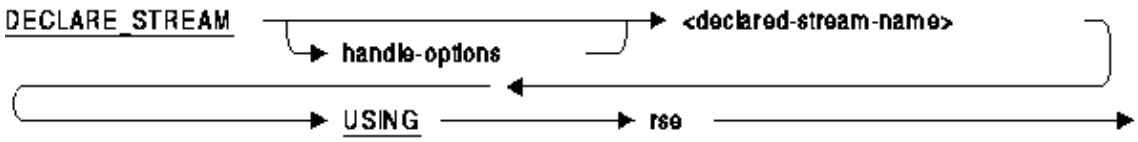
This release note was included in the V70A Release Notes but had gotten dropped somewhere along the line.

For this release of Rdb, two new keywords have been added to the handle−options for the DECLARE_STREAM, the START_STREAM (undeclared format) and FOR loop statements. These changes have been made to RDBPRE, RDO and RDB$INTERPRET at the request of several RDO customers.

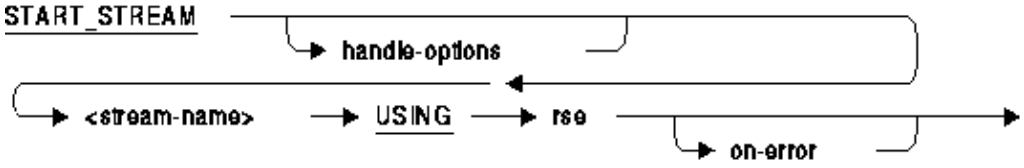
In prior releases, the handle−options could not be specified in interactive RDO or RDB$INTERPRET. This has changed in Rdb but these allowed options will be limited to MODIFY and PROTECTED keywords. For RDBPRE, all options listed will be supported. These option names were chosen to be existing keywords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

The altered statements are shown in Example 5−1, Example 5−2 and Example 5−3.

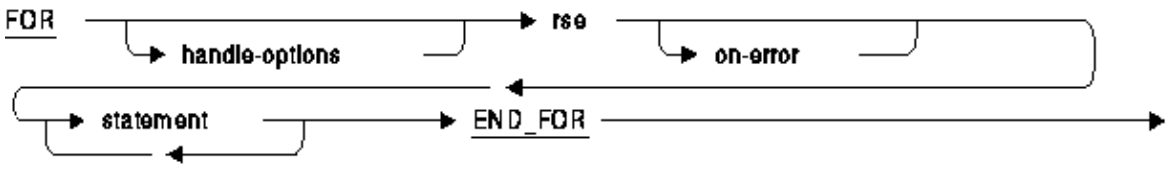
Example 5−1 DECLARE_STREAM Format



Example 5−2 START_STREAM Format

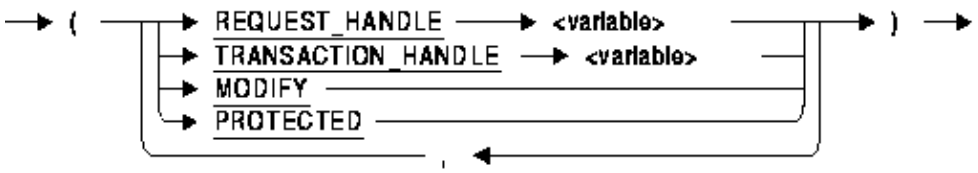


Example 5−3 FOR Format



Each of these statements references the syntax for the HANDLE−OPTIONS which has been revised and is shown below.

handle-options =



The following options are available for HANDLE−OPTIONS:

- REQUEST_HANDLE specifies the request handle for this request. This option is only valid for RDBPRE and RDML applications. It cannot be used with RDB$INTERPRET, nor interactive RDO.
- TRANSACTION_HANDLE specifies the transaction handle under which this request executes. This option is only valid for RDBPRE and RDML applications. It cannot be used with RDB$INTERPRET, nor interactive RDO.
- MODIFY specifies that the application will modify all (or most) records fetched from the stream or for loop. This option can be used to improve application performance by avoiding lock promotion from SHARED READ for the FETCH to PROTECTED WRITE access for the nested MODIFY or ERASE statement. It can also reduce DEADLOCK occurrence because lock promotions are avoided. This option is valid for RDBPRE, RDB$INTERPRET, and interactive RDO. This option is not currently available for RDML.
  For example:

```
RDO>   FOR (MODIFY) E IN EMPLOYEES WITH E.EMPLOYEE_ID = "00164"
cont>    MODIFY E USING E.MIDDLE_INITIAL = "M"
cont>    END_MODIFY
cont>  END_FOR
```

  This FOR loop uses the MODIFY option to indicate that the nested MODIFY is an unconditional statement and so aggressive locking can be undertaken during the fetch of the record in the FOR loop.
- PROTECTED specifies that the application may modify records fetched by this stream by a separate and independent MODIFY statement. Therefore, this stream should be protected from interference (aka Halloween affect). The optimizer will select a snapshot of the rows and store them in a temporary relation for processing, rather than traversing indexes at the time of the FETCH statement. In some cases this may result in poorer performance when the temporary relation is large and overflows from virtual memory to a temporary disk file, but the record stream will be protected from interference. The programmer is directed to the documentation for the Oracle Rdb logical names

RDMS$BIND_WORK_VM and RDMS$BIND_WORK_FILE.
This option is valid for RDBPRE, RDB$INTERPRET, and interactive RDO. This option is not currently available for RDML.
The following example creates a record stream in a BASIC program using Callable RDO:

```
RDMS_STATUS = RDB$INTERPRET ('INVOKE DATABASE PATHNAME "PERSONNEL"')

RDMS_STATUS = RDB$INTERPRET ('START_STREAM (PROTECTED) EMP USING ' + &
                             'E IN EMPLOYEES')

RDMS_STATUS = RDB$INTERPRET ('FETCH EMP')

DML_STRING = 'GET ' +                                          &
                 '!VAL = E.EMPLOYEE_ID;' +                     &
                 '!VAL = E.LAST_NAME;' +                       &
                 '!VAL = E.FIRST_NAME' +                       &
             'END_GET'

RDMS_STATUS = RDB$INTERPRET (DML_STRING, EMP_ID, LAST_NAME, FIRST_NAME)
```

In this case the FETCH needs to be protected against MODIFY statements which execute in other parts of the application.

7.1.7 Missing Descriptions of RDB$FLAGS from HELP File

The HELP file for Oracle Rdb Release 7.0 describes the system tables for Oracle Rdb and was missing these updated descriptions of the RDB$FLAGS column for several tables.

Table 7−3 Changed Columns for RDB$INDICES Table

Column Name	Data Type	Domain Name	Comments
RDB$FLAGS	Integer	RDB$FLAGS	A bit mask where the bits have the following meaning when set:
			Bit 0: This index is of type HASHED.
			Bit 1: This index uses the MAPPING VALUES clause to compress ainteger value ranges.
			Bit 2: If this is a HASHED index then it is of type ORDERED. If clear this indicates the index is of type SCATTERED.
			Bit 3: Reserved for future use.
			Bit 4: This index has run length compression enabled (ENABLE COMPRESSION).
			Bit 5: This index is no longer used (MAINTENANCE IS DISABLED).
			Bit 6 through 10: Reserved for future use.
			Bit 11: This index has duplicates compressed (DUPLICATES ARE COMPRESSED).

Bit 12: This index is of type SORTED RANKED.
Bits 13 through 31: Reserved for future use.

Table 7−4 Changed Columns for Rdb$RELATIONS Table

Column Name	Data Type	Domain Name	Comments
RDB$FLAGS	Integer	RDB$FLAGS	A bit mask where the bits have the following meaning when set:
			Bit 0: This relation is a view.
			Bit 1: This relation is not compressed.
			Bit 2: The SQL clause, WITH CHECK OPTION, is used in this view definition.
			Bit 3: Indicates a special internal system relation.
			Bit 4: This view is not an ANSI updatable view.
			Bit 5: This is an imported table in the Distributed Option for Rdb catalog.
			Bit 6: This is a passthru table in the Distributed Option for Rdb catalog.
			Bit 7: This is a partitioned view in the Distributed Option for Rdb catalog.
			Bit 8: This table has compression defined by the storage map. When set Bit 1 in this bit mask is ignored.
			Bit 9: This is a temporary table.
			Bit 10: When bit 9 is set this is a global temporary table, when clear it indicates a local temporary table.
			Bit 11: When bit 9 is set this indicates that the rows in the temporary table should be deleted upon COMMIT.
			Bit 12: Reserved for future use.
			Bit 13: A table (via a computed by column) or view references a local temporary table.
			Bit 14: Reserved for future use.
			Bit 15: This is a system table with a special storage map.
			Bits 16 through 31: Reserved for future use.

Table 7−5 Changed Columns for RDB$ROUTINES Table

Column Name	Data Type	Domain Name	Comments
RDB$FLAGS	Integer	RDB$FLAGS	A bit mask where the bits have the following meaning when set:
			Bit 0: Routine is a function. (call returns a result.)
			Bit 1: Routine is not valid. (Invalidated by a metadata change.)
			Bit 2: Function is not deterministic (that is, the routine is variant). A subsequent invocation of the routine with identical parameters

may return different results.
Bit 3: Routine can change the transaction state.
Bit 4: Routine is in a secured shareable image.
Bit 5: Reserved for future use.
Bit 6: Routine is not valid. (Invalidated by a metadata change to the object upon which this routine depends. This dependency is a language semantics dependency.)
Bit 7: Reserved for future use.
Bit 8: External function returns NULL when called with any NULL parameter.
Bit 9: Routine has been analyzed (used for trigger dependency tracking).
Bit 10: Routine inserts rows.
Bit 11: Routine modifies rows.
Bit 12: Routine deletes rows.
Bit 13: Routine selects rows.
Bit 14: Routine calls other routines.
Other bits are reserved for future use.

Table 7−6 Changed Columns for RDB$STORAGE_MAPS Table

Column Name	Data Type	Domain Name	Comments
RDB$FLAGS	Integer	RDB$FLAGS	A bit mask where the bits have the following meaning when set:
			Bit 0: This table or index is mapped to page format MIXED areas.
			Bit 1: This partition is not compressed.
			Bit 2: This is a strictly partitioned storage map, the partitioning columns become read only for UPDATE.
			Bit 3 through 31: Reserved for future use.

7.1.8 A Way to Find the Transaction Type of a Particular Transaction Within the Trace Database

The table EPC$1_221_TRANSACTION in the formatted Oracle Trace database has a column LOCK_MODE_START of longword datatype. The values of this column indicate the type of transaction a particular transaction was.

```
Value           Transaction type
-----           ----------------
8               Read only
9               Read write
14              Batch update
```

7.1.9 Clarification of SET FLAGS Option DATABASE_PARAMETERS

Bug 1668049

The Oracle Rdb7 SQL Reference Manual describes the option DATABASE_PARAMETERS in Table 7−6 in the SET FLAGS section. However, this keyword generates output only during ATTACH to the database which happens prior to the SET FLAGS statement executing.

This option is therefore only useful when used with the RDMS$SET_FLAGS logical name which provides similar functionality.

```
$ define RDMS$SET_FLAGS "database_parameters"
$ sql$
SQL> Attach 'File db$:scratch';
 ATTACH #1, Database BLUGUM$DKA300:[SMITHI.DATABASES.V70]SCRATCH.RDB;1
~P Database Parameter Buffer (version=2, len=79)
0000 (00000) RDB$K_DPB_VERSION2
0001 (00001) RDB$K_FACILITY_ALL
0002 (00002) RDB$K_DPB2_IMAGE_NAME "NODE::DISK:[DIR]SQL$70.EXE;1"
0040 (00064) RDB$K_FACILITY_ALL
0041 (00065) RDB$K_DPB2_DBKEY_SCOPE (Transaction)
0045 (00069) RDB$K_FACILITY_ALL
0046 (00070) RDB$K_DPB2_REQUEST_SCOPE (Attach)
004A (00074) RDB$K_FACILITY_RDB_VMS
004B (00075) RDB$K_DPB2_CDD_MAINTAINED (No)
 RDMS$BIND_WORK_FILE = "DISK:[DIR]RDMSTTBL$UEOU3LQ0RV2.TMP;" (Visible = 0)
SQL> Exit
 DETACH #1
```

7.1.10 Additional Information About Detached Processes

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

Applications run from a detached process must ensure that the OpenVMS environment is established correctly before running Oracle Rdb. Otherwise, Oracle Rdb will not execute.

Attempts to attach to a database and execute an Oracle Rdb query from applications running as detached processes will result in an error similar to the following:

```
%RDB−F−SYS_REQUEST, error from system services request
−SORT−E−OPENOUT, error opening {file} as output
−RMS−F−DEV, error in device name or inappropriate device type for
operation
```

The problem occurs because a detached process does not normally have the logical names SYS$LOGIN or SYS$SCRATCH defined.

There are two methods that can be used to correct this:

1. Use the DCL command procedure RUN_PROCEDURE to run the ACCOUNTS application: RUN_PROCEDURE.COM includes the single line:

```
$ RUN ACCOUNTS_REPORT
```

Then execute this procedure using this command:

```
$ RUN/DETACH/AUTHORIZE SYS$SYSTEM:LOGINOUT/INPUT=RUN_PROCEDURE
```

This solution executes SYS$SYSTEM:LOGINOUT so that the command language interface (DCL) is activated. This causes the logical names SYS$LOGIN and SYS$SCRATCH to be defined for the detached process. The /AUTHORIZE qualifier also ensures that the users' process quota limits (PQLs) are used from the system authorization file rather than relying on the default PQL system parameters, which are often insufficient to run Oracle Rdb.

2. If DCL is not desired, and SYS$LOGIN and SYS$SCRATCH are not defined, then prior to executing any Oracle Rdb statement, you should define the following logical names:
   - RDMS$BIND_WORK_FILE
     Define this logical name to allow you to reduce the overhead of disk I/O operations for matching operations when used in conjunction with the RDMS$BIND_WORK_VM logical name. If the virtual memory file is too small, then overflow to disk will occur at the disk and directory location specified by RDMS$BIND_WORK_FILE.
     For more information on RDMS$BIND_WORK_FILE and RDMS$BIND_WORK_VM, see the Oracle Rdb Guide to Database Performance and Tuning.
   - SORTWORK0, SORTWORK1, and so on
     The OpenVMS sort/merge utility (SORT/MERGE) attempts to create sort work files in SYS$SCRATCH. If the SORTWORK logical names exist, the utility will not require the SYS$SCRATCH logical. However, note that not all queries will require sorting, and that some sorts will be completed in memory and so will not necessarily require disk space.
     If you use the logical RDMS$BIND_SORT_WORKFILES, you will need to define further SORTWORK logical names as described in the Oracle Rdb Guide to Database Performance and Tuning.

You should also verify that sufficient process quotas are specified on the RUN/DETACH command line, or defined as system PQL parameters to allow Oracle Rdb to execute.

7.1.11 The Halloween Problem

When a cursor is processing rows selected from a table, it is possible that another separate query can interfere with the retrieval of the cursor by modifying the index column's key values used by the cursor.

For instance, if a cursor selects all EMPLOYEES with LAST_NAME >= 'M', it is likely that the query will use the sorted index on LAST_NAME to retrieve the rows for the cursor. If an update occurs during the processing of the cursor which changes the LAST_NAME of an employee from "Mason" to "Rickard", then it is possible that that employee row will be processed twice. First, when it is fetched with name "Mason", and then later when it is accessed by the new name "Rickard".

The Halloween problem is a well known problem in relational databases. Access strategies which optimize the I/O requirements, such as Index Retrieval, can be subject to this problem. Interference from queries by other sessions are avoided by locking and are controlled by the ISOLATION LEVEL options in SQL, or the CONCURRENCY/CONSISTENCY options in RDO/RDML.

Oracle Rdb avoids this problem if it knows that the cursors subject table will be updated. For example, if the SQL syntax UPDATE ... WHERE CURRENT OF is used to perform updates of target rows, or if the RDO/RDML MODIFY statement uses the context variable for the stream. Then the optimizer will choose an alternate access strategy if an update can occur which may cause the Halloween problem. This can be seen in the access strategy in the example below as a "Temporary relation" being created to hold the result of the cursor query.

When you use interactive or dynamic SQL, the UPDATE ... WHERE CURRENT OF or DELETE ... WHERE CURRENT OF statements will not be seen until after the cursor is declared and opened. In these environments, you must use the FOR UPDATE clause to specify that columns selected by the cursor will be updated during cursor processing. This is an indication to the Rdb optimizer so that it protects against the Halloween problem in this case. This is shown in the two examples below Example 7−1 and Example 7−2.

Example 7−1 shows that the EMP_LAST_NAME index is used for retrieval. Any update performed will possibly be subject to the Halloween problem.

Example 7−1 Interactive Cursor with no Halloween Protection

```
SQL> set flags 'strategy';
SQL> declare emp  cursor for
cont> select * from employees where last_name >= 'M'
cont> order by last_name;
SQL> open emp;
Conjunct        Get     Retrieval by index of relation EMPLOYEES
  Index name  EMP_LAST_NAME [1:0]
SQL> close emp;
```

Example 7−2 shows that the query specifies that the column LAST_NAME will be updated by some later query. Now the optimizer protects the EMP_LAST_NAME index used for retrieval by using a "Temporary Relation" to hold the query result set. Any update performed on LAST_NAME will now avoid the Halloween problem.

Example 7−2 Interactive Cursor with Halloween Protection

```
SQL> set flags 'strategy';
SQL> declare emp2 cursor for
cont> select * from employees where last_name >= 'M'
cont> order by last_name
cont> for update of last_name;
SQL> open emp2;
Temporary relation      Conjunct        Get
Retrieval by index of relation EMPLOYEES
  Index name  EMP_LAST_NAME [1:0]
SQL> close emp2;
```

When you use the SQL precompiler or the SQL module language compiler, it can be determined from usage that the cursor context will possibly be updated during the processing of the cursor because all cursor related statements are present within the module. This is also true for the RDML/RDBPRE precompilers when you use the DECLARE_STREAM and START_STREAM statements and use the same stream context to perform all MODIFY and ERASE statements.

The point to note here is that the protection takes place during the open of the SQL cursor (or RDO

stream), not during the subsequent UPDATE or DELETE.

If you execute a separate UPDATE query which modifies rows being fetched from the cursor, then the actual rows fetched will depend upon the access strategy chosen by the Rdb optimizer. As the query is separate from the cursors query (i.e. doesn't reference the cursor context) then the optimizer does not know that the cursor selected rows are potentially updated and so can not perform the normal protection against the Halloween problem.

7.1.12 RDM$BIND_MAX_DBR_COUNT Documentation Clarification

Bug 1495227

The Rdb7 Guide to Database Performance and Tuning Manual, Volume 2, Page A−18, incorrectly describes the use of the RDM$BIND_MAX_DBR_COUNT logical.

Following is an updated description. Note that the difference in actual behavior between what is in the existing documentation and software is that the logical name only controls the number of database recovery processes created at once during "node failure" recovery (that is, after a system or monitor crash or other abnormal shutdown).

When an entire database is abnormally shut down (due, for example, to a system failure), the database will have to be recovered in a "node failure" recovery mode. This recovery will be performed by another monitor in the cluster if the database is opened on another node or will be performed the next time the database is opened.

The RDM$BIND_MAX_DBR_COUNT logical name and the RDB_BIND_MAX_DBR_COUNT configuration parameter define the maximum number of database recovery (DBR) processes to be simultaneously invoked by the database monitor during a "node failure" recovery.

This logical name and configuration parameter apply only to databases that do not have global buffers enabled. Databases that utilize global buffers have only one recovery process started at a time during a "node failure" recovery.

In a "node failure" recovery situation with the Row Cache feature enabled (regardless of the global buffer state), the database monitor will start a single database recovery (DBR) process to recover the Row Cache Server (RCS) process and all user processes from the oldest active checkpoint in the database.

7.1.13 RMU /UNLOAD /AFTER_JOURNAL NULL Bit Vector Clarification

Each output record from the RMU /UNLOAD /AFTER_JOURNAL command includes a vector (array) of bits. There is one bit for each field in the data record. If a null bit value is 1, the corresponding field is NULL; if a null bit value is 0, the corresponding field is not NULL and contains an actual data value. The contents of a data field that is NULL are not initialized and are not predictable.

The null bit vector begins on a byte boundary. The field RDB$LM_NBV_LEN indicates the number of valid bits (and thus, the number of columns in the table). Any extra bits in the final byte of the vector after the final null bit are unused and the contents are unpredictable.

The following example C program demonstrates one possible way of reading and parsing a binary output file (including the null bit vector) from the RMU /UNLOAD /AFTER_JOURNAL command. This sample program has been tested using Oracle Rdb V7.0.5 and HP C V6.2−009 on OpenVMS Alpha V7.2−1. It is meant to be used as a template for writing your own program.

```
/* DATATYPES.C */

#include <stdio.h>
#include <descrip.h>
#include <starlet.h>
#include <string.h>

#pragma member_alignment __save
#pragma nomember_alignment

struct { /* Database key structure */
    unsigned short      lno;    /* line number */
    unsigned int        pno;    /* page number */
    unsigned short      dbid;   /* area number */
    } dbkey;

typedef struct { /* Null bit vector with one bit for each column */
    unsigned            n_tinyint   :1;
    unsigned            n_smallint  :1;
    unsigned            n_integer   :1;
    unsigned            n_bigint    :1;
    unsigned            n_double    :1;
    unsigned            n_real      :1;
    unsigned            n_fixstr    :1;
    unsigned            n_varstr    :1;
    } nbv_t;

struct { /* LogMiner output record structure for table DATATYPES */
    char                rdb$lm_action;
    char                rdb$lm_relation_name [31];
    int                 rdb$lm_record_type;
    short               rdb$lm_data_len;
    short               rdb$lm_nbv_len;
    __int64             rdb$lm_dbk;
    __int64             rdb$lm_start_tad;
    __int64             rdb$lm_commit_tad;
    __int64             rdb$lm_tsn;
    short               rdb$lm_record_version;
    char                f_tinyint;
    short               f_smallint;
    int                 f_integer;
    __int64             f_bigint;
    double              f_double;
    float               f_real;
    char                f_fixstr[10];
    short               f_varstr_len;   /* length of varchar */
    char                f_varstr[10];   /* data of varchar */
    nbv_t               nbv;
    } lm;

#pragma member_alignment __restore
```

```
main ()
{   char timbuf [24];
    struct dsc$descriptor_s dsc = {
        23, DSC$K_DTYPE_T, DSC$K_CLASS_S, timbuf};
    FILE *fp = fopen ("datatypes.dat", "r", "ctx=bin");


    memset (&timbuf, 0, sizeof(timbuf));

    while (fread (&lm, sizeof(lm), 1, fp) != 0)
    {
        printf ("Action     = %c\n",     lm.rdb$lm_action);
        printf ("Table      = %.*s\n",   sizeof(lm.rdb$lm_relation_name),
                                         lm.rdb$lm_relation_name);
        printf ("Type       = %d\n",     lm.rdb$lm_record_type);
        printf ("Data Len   = %d\n",     lm.rdb$lm_data_len);
        printf ("Null Bits  = %d\n",     lm.rdb$lm_nbv_len);

        memcpy (&dbkey, &lm.rdb$lm_dbk, sizeof(lm.rdb$lm_dbk));
        printf ("DBKEY      = %d:%d:%d\n", dbkey.dbid,
                                           dbkey.pno,
                                           dbkey.lno);

        sys$asctim (0, &dsc, &lm.rdb$lm_start_tad, 0);
        printf ("Start TAD  = %s\n", timbuf);

        sys$asctim (0, &dsc, &lm.rdb$lm_commit_tad, 0);
        printf ("Commit TAD = %s\n", timbuf);

        printf ("TSN        = %Ld\n",    lm.rdb$lm_tsn);
        printf ("Version    = %d\n",     lm.rdb$lm_record_version);

        if (lm.nbv.n_tinyint == 0)
                printf ("f_tinyint  = %d\n", lm.f_tinyint);
        else    printf ("f_tinyint  = NULL\n");

        if (lm.nbv.n_smallint == 0)
                printf ("f_smallint = %d\n", lm.f_smallint);
        else    printf ("f_smallint = NULL\n");

        if (lm.nbv.n_integer == 0)
                printf ("f_integer  = %d\n", lm.f_integer);
        else    printf ("f_integer  = NULL\n");

        if (lm.nbv.n_bigint == 0)
                printf ("f_bigint   = %Ld\n", lm.f_bigint);
        else    printf ("f_bigint   = NULL\n");

        if (lm.nbv.n_double == 0)
                printf ("f_double   = %f\n", lm.f_double);
        else    printf ("f_double   = NULL\n");

        if (lm.nbv.n_real == 0)
                printf ("f_real     = %f\n", lm.f_real);
        else    printf ("f_real     = NULL\n");

        if (lm.nbv.n_fixstr == 0)
                printf ("f_fixstr   = %.*s\n", sizeof (lm.f_fixstr),
                                                 lm.f_fixstr);
        else    printf ("f_fixstr   = NULL\n");
```

```
        if (lm.nbv.n_varstr == 0)
                printf ("f_varstr   = %.*s\n", lm.f_varstr_len, lm.f_varstr);
        else    printf ("f_varstr   = NULL\n");

        printf ("\n");
    }
}
```

Example sequence of commands to create a table, unload the data and display the contents with this program:

```
SQL> ATTACH 'FILE MF_PERSONNEL';
SQL> CREATE TABLE DATATYPES (
     F_TINYINT TINYINT
    ,F_SMALLINT SMALLINT
    ,F_INTEGER INTEGER
    ,F_BIGINT BIGINT
    ,F_DOUBLE DOUBLE PRECISION
    ,F_REAL REAL
    ,F_FIXSTR CHAR (10)
    ,F_VARSTR VARCHAR (10));
SQL> COMMIT;
SQL> INSERT INTO DATATYPES VALUES (1, NULL, 2, NULL, 3, NULL, 'THIS', NULL);
SQL> INSERT INTO DATATYPES VALUES (NULL, 4, NULL, 5, NULL, 6, NULL, 'THAT');
SQL> COMMIT;
SQL> EXIT;
$ RMU /BACKUP /AFTER_JOURNAL MF_PERSONNEL AIJBCK.AIJ
$ RMU /UNLOAD /AFTER_JOURNAL MF_PERSONNEL AIJBCK.AIJ -
    /TABLE = (NAME=DATATYPES, OUTPUT=DATATYPES.DAT)
$ CC DATATYPES.C
$ LINK DATATYPES.OBJ
$ RUN DATATYPES.EXE
```

7.1.14 Location of Host Source File Generated by the SQL Precompilers

Bug 478898

When the SQL precompiler generates host source files (like .c, .pas, .for) from the precompiler source files, it locates these files based on the /obj qualifier located on the command line given to the SQL precompiler.

The following examples show the location where the host source file is generated. When /obj is not specified on the command line, the object and the host source file take the name of the SQL precompiler source files with the extensions of .obj and .c respectively.

```
LUND> sqlpre/cc scc_try_mli_successful.sc
LUND> dir scc_try_mli_successful.*

Directory MYDISK:[LUND]

SCC_TRY_MLI_SUCCESSFUL.C;1              SCC_TRY_MLI_SUCCESSFUL.OBJ;2
SCC_TRY_MLI_SUCCESSFUL.SC;2

Total of 3 files.
```

When /obj is specified on the command line, the object and the host source take the name given on the qualifier switch. It uses the default of the SQL precompiler source if a filespec is not specified. It uses the defaults of .obj and .c if the extension is not specified. If the host language is other than C, then it uses the appropriate host source extension (like .pas, .for, etc). The files also default to the current directory if a directory specification is not specified.

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

Directory MYDISK:[LUND]

SCC_TRY_MLI_SUCCESSFUL.SC;2

Total of 1 file.
LUND> dir myobj.*

Directory MYDISK:[LUND]

MYOBJ.C;1           MYOBJ.OBJ;2

Total of 2 files.

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

Directory MYDISK:[LUND]

SCC_TRY_MLI_SUCCESSFUL.SC;2

Total of 1 file.
LUND> dir MYDISK:[lund.tmp]scc_try_mli_successful.*

Directory MYDISK:[LUND.TMP]

SCC_TRY_MLI_SUCCESSFUL.C;1              SCC_TRY_MLI_SUCCESSFUL.OBJ;2

Total of 2 files.
```

This problem has been corrected in Oracle Rdb Release 7.0.6.

7.1.15 Suggestion to Increase GH_RSRVPGCNT Removed

The Oracle Rdb7 for OpenVMS Installation and Configuration Guide contains a section titled "Installing Oracle Rdb Images as Resident on OpenVMS Alpha" that includes information about increasing the value of the OpenVMS system parameter GH_RSRVPGCNT when you modify the RMONSTART.COM or SQL$STARTUP.COM procedures to install Rdb images with the /RESIDENT qualifier.

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

parameter.

Oracle and HP suggest that you do not modify the value of the GH_RSRVPGCNT system parameter unless it is absolutely required. Some versions of OpenVMS on some hardware platforms require that GH_RSRVPGCNT be zero in order to ensure the highest level of system performance.

7.1.16 Clarification of the DDLDONOTMIX Error Message

Bug 454080

The ALTER DATABASE statement performs two classes of functions: changing the database root structures in the .RDB file and modifying the system metadata in the RDB$SYSTEM storage area. The first class of changes do not require a transaction to be active. However, the second class requires that a transaction be active. Oracle Rdb does not currently support the mixing of these two classes of ALTER DATABASE clauses.

When you mix clauses that fall into both classes, the error message DDLDONOTMIX "the {SQL−syntax} clause can not be used with some ALTER DATABASE clauses" is displayed, and the ALTER DATABASE statement fails.

```
SQL> alter database filename MF_PERSONNEL
cont> dictionary is not used
cont> add storage area JOB_EXTRA filename JOB_EXTRA;
%RDB−F−BAD_DPB_CONTENT, invalid database parameters in the database parameter
block (DPB)
−RDMS−E−DDLDONOTMIX, the "DICTIONARY IS NOT USED" clause can not be used with
some ALTER DATABASE clauses
```

The following clauses may be mixed with each other but may not appear with other clauses such as ADD STORAGE AREA, or ADD CACHE.

- DICTIONARY IS [NOT] REQUIRED
- DICTIONARY IS NOT USED
- MULTISCHEMA IS { ON | OFF }
- CARDINALITY COLLECTION IS { ENABLED | DISABLED }
- METADATA CHANGES ARE { ENABLED | DISABLED }
- WORKLOAD COLLECTION IS { ENABLED | DISABLED }

If the DDLDONOTMIX error is displayed, then restructure the ALTER DATABASE into two statements, one for each class of actions.

```
SQL> alter database filename MF_PERSONNEL
cont> dictionary is not used;
SQL> alter database filename MF_PERSONNEL
cont> add storage area JOB_EXTRA filename JOB_EXTRA;
```

7.1.17 Compressed Sorted Index Entry Stored in Incorrect Storage Area

This note was originally included in the Oracle Rdb Release 7.0.1.3 and 7.0.2 Release Notes. The logical name documented in the note for those releases was documented incorrectly. Below is a corrected note.

In specific cases, in versions V6.1 and V7.0 of Oracle Rdb, when a partitioned, compressed sorted index was created after the data was inserted into the table, b−tree entries may have been inserted into the wrong storage area.

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

- CREATE INDEX is issued after there are records already in the table on which the index is being created
- index must be partitioned over a single column
- index must have compression enabled
- scale factor must be zero on the columns of the index
- no collating sequences specified on the columns of the index
- no descending indexes
- MAPPING VALUES must not be specified

RMU/DUMP/AREA=xx will show that the b−tree entry was not stored in the expected storage area. However, in versions V6.1 and V7.0 of Oracle Rdb, the rows of the table can still be successfully retrieved.

The following example shows the problem:

```
create database
    filename foo
 create storage area Area_1
        filename Area_1
 create storage area Area_2
        filename Area2;

create table T1
        (C1 integer);

! insert data into table prior to index creation
insert into T1 values (0);
commit;

! create index with COMPRESSION ENABLED
create index Index_1
    on T1 (C1)
    enable compression
    store using (C1)
        in Area_1 with limit of (0)
        otherwise in Area_2;
COMMIT;
!
! Dump out the page for b−tree in AREA_1, there are 0 bytes stored.
! There should be 5 bytes stored for the b−tree entry.
!
RMU/DUMP/AREA=AREA_1
.
.
.                                ....  total B−tree node size: 430
                       0030 2003  0240  line 0 (2:5:0) index: set 48
              002F FFFFFFFF FFFF  0244  owner 47:−1:−1
                            0000  024C  0 bytes of entries <−−−***** no entry
```

```
                              8200  024E  level 1, full suffix
00000000000000000000000000000000  0250  unused '................'
.
.
.
!
! Dump out the page for b-tree in AREA_2, there are 5 bytes stored
!
RMU/DUMP/AREA=AREA_2
.
.
.                                 ....  total B-tree node size: 430
                        0031 2003  0240  line 0 (3:5:0) index: set 49
                002F FFFFFFFF FFFF  0244  owner 47:-1:-1
                              000A  024C  10 bytes of entries
                              8200  024E  level 1, full suffix
                             00 05  0250  5 bytes stored, 0 byte prefix <---entry
                         0100008000  0252   key '.....'
                            22B1 10  0257   pointer 47:554:0
.
.
.
```

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

Note

In prior versions, the rows were successfully retrieved even though the key values were stored in the wrong storage area. This was due to the range query algorithm skipping empty partitions or scanning extra areas.

However, due to an enhancement in the algorithm for range queries on partitioned SORTED indexes in Oracle Rdb Release 7.0.2, the rows of the table which are stored in the incorrect storage areas may not be retrieved when using the partitioned index.

The optimized algorithm now only scans the relevant index areas (and no longer skips over empty areas) resulting in only those rows being returned. Therefore, it is recommended that the index be dropped and re−created. For a short term solution, another alternative is to disable the new optimization by defining the logical RDMS$INDEX_PART_CHECK to 0.

This problem has been corrected in Oracle Rdb Release 7.0.1.3.

7.1.18 Partition Clause is Optional on CREATE STORAGE MAP

Bug 642158

In the *Oracle Rdb7 SQL Reference Manual*, the syntax diagram for the CREATE STORAGE MAP

statement incorrectly shows the partition clause as required syntax. The partition clause is not a required clause.

This correction will appear in the next publication of the *Oracle Rdb SQL Reference Manual*.

7.1.19 Oracle Rdb Logical Names

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

Logical Name Configuration Parameter	Function
RDM$BIND_RUJ_ALLOC_BLKCNT	Allows you to override the default value of the .ruj file. The block count value can be defined between 0 and 2 billion with a default of 127.
RDM$BIND_RUJ_EXTEND_BLKCNT	Allows you to pre−extend the .ruj files for each process using a database. The block count value can be defined between 0 and 65535 with a default of 127.

7.1.20 Documentation Error in *Oracle Rdb Guide to Database Performance and Tuning*

The *Oracle Rdb7 Guide to Database Performance and Tuning, Volume 2* contains an error in section C.7, "Displaying Sort Statistics with the R Flag".

When describing the output from this debugging flag, bullet 9 states:

- ***Work File Alloc*** indicates how many work files were used in the sort operation. A zero (0) value indicates that the sort was accomplished completely in memory.

This is incorrect. This statistic should be described as shown:

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

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

7.1.21 SET FLAGS Option IGNORE_OUTLINE Not Available

Bug 510968

The *Oracle Rdb7 SQL Reference Manual* described the option IGNORE_OUTLINE in Table 7−6 of the SET FLAGS section. However, this keyword was not implemented in Oracle Rdb Release 7.0.

This has been corrected in this release of Oracle Rdb. This keyword is now recognized by the SET FLAGS statement. As a workaround the logical name RDMS$BIND_OUTLINE_FLAGS "I" can be used to set this attribute.

7.1.22 SET FLAGS Option INTERNALS Not Described

The *Oracle Rdb7 SQL Reference Manual* does not describe the option INTERNALS in Table 7−6 in the SET FLAGS section. This keyword was available in first release of Oracle Rdb 7.0 and is used to enable debug flags output for internal queries such as constraints and triggers. It can be used in conjunction with other options such as STRATEGY, BLR, and EXECUTION. For example, the following flag settings are equivalent to defining the RDMS$DEBUG_FLAGS as ISn and shows the strategy used by the trigger's actions on the AFTER DELETE trigger on the EMPLOYEES table.

```
SQL> SET FLAGS 'STRATEGY, INTERNAL, REQUEST_NAME';
SQL> SHOW FLAGS

Alias RDB$DBHANDLE:
Flags currently set for Oracle Rdb:
   INTERNALS,STRATEGY,PREFIX,REQUEST_NAMES
SQL> DELETE FROM EMPLOYEES WHERE EMPLOYEE_ID = '00164';
~S: Trigger name  EMPLOYEE_ID_CASCADE_DELETE
Get     Temporary relation      Retrieval by index of relation DEGREES
  Index name  DEG_EMP_ID [1:1]
~S: Trigger name  EMPLOYEE_ID_CASCADE_DELETE
Get     Temporary relation      Retrieval by index of relation JOB_HISTORY
  Index name  JOB_HISTORY_HASH [1:1]
~S: Trigger name  EMPLOYEE_ID_CASCADE_DELETE
Get     Temporary relation      Retrieval by index of relation SALARY_HISTORY
  Index name  SH_EMPLOYEE_ID [1:1]
~S: Trigger name  EMPLOYEE_ID_CASCADE_DELETE
Conjunct        Get     Retrieval by index of relation DEPARTMENTS
  Index name  DEPARTMENTS_INDEX [0:0]
Temporary relation      Get     Retrieval by index of relation EMPLOYEES
  Index name  EMPLOYEES_HASH [1:1]       Direct lookup
1 row deleted
```

7.1.23 Documentation for VALIDATE_ROUTINE Keyword for SET FLAGS

The SET FLAGS section of the *Oracle Rdb7 SQL Reference Manual* omitted the description of the VALIDATE_ROUTINE keyword (which can be negated as NOVALIDATE_ROUTINE). This keyword enables the re−validation of an invalidated stored procedure or function. This flag has the same action as the logical RDMS$VALIDATE_ROUTINE described in the *Oracle Rdb7 Guide to Database Performance and Tuning*.

This example shows the re−validation of a stored procedure. When the stored routine is successfully prepared (but not executed), the setting of VALIDATE_ROUTINE causes the entry for this routine in the RDB$ROUTINES system table to be set as valid.

```
SQL> SET TRANSACTION READ WRITE;
SQL> SET FLAGS 'VALIDATE_ROUTINE';
SQL> SET NOEXECUTE;
SQL> CALL ADD_EMPLOYEE ('Smith');
```

```
SQL> SET EXECUTE;
SQL> COMMIT;
```

In this example, the use of the SET NOEXECUTE statement in interactive SQL allows the stored routine to be successfully compiled, but it is not executed.

7.1.24 Documentation for Defining the RDBSERVER Logical Name

Bugs 460611 and 563649.

Sections 4.3.7.1 and 4.3.7.2 in the *Oracle Rdb7 for OpenVMS Installation and Configuration Guide* provide the following examples for defining the RDBSERVER logical name:

```
 $ DEFINE RDBSERVER SYS$SYSTEM:RDBSERVER70.EXE
  and
 $ DEFINE RDBSERVER SYS$SYSTEM:RDBSERVER61.EXE
```

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

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

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

The *Oracle Rdb7 for OpenVMS Installation and Configuration Guide* should define the logical name as follows:

```
 DEFINE RDBSERVER SYS$COMMON:<SYSEXE>RDBSERVER70.EXE
  and
 DEFINE RDBSERVER SYS$COMMON:<SYSEXE>RDBSERVER61.EXE
```

7.1.25 Undocumented SET Commands and Language Options

The following SET statements were omitted from the Oracle Rdb7 documentation.

7.1.25.1 QUIET COMMIT Option

The SET QUIET COMMIT statement (for interactive and dynamic SQL), the module header option QUIET COMMIT, the /QUIET_COMMIT (and /NOQUIET_COMMIT) qualifier for SQL module language, or the /SQLOPTIONS=QUIET_COMMIT (and NOQUIET_COMMIT) option for the SQL language precompiler allows the programmer to control the behavior of the COMMIT and ROLLBACK statements in cases where there is no active transaction.

By default, if there is no active transaction, SQL will raise an error when COMMIT or ROLLBACK is executed. This default is retained for backward compatibility for applications that may wish to detect the situation. If QUIET COMMIT is set to ON, then a COMMIT or ROLLBACK executes successfully when there is no active transaction.

Note

Within a compound statement, the COMMIT and ROLLBACK statements in this case are ignored.

Examples

In interactive or dynamic SQL, the following SET command can be used to disable or enable error reporting for COMMIT and ROLLBACK when no transaction is active. The parameter to the SET command is a string literal or host variable containing the keyword ON or OFF. The keywords may be in any case (upper, lower, or mixed).

```
SQL> COMMIT;
%SQL-F-NO_TXNOUT, No transaction outstanding
SQL> ROLLBACK;
%SQL-F-NO_TXNOUT, No transaction outstanding
SQL> SET QUIET COMMIT 'on';
SQL> ROLLBACK;
SQL> COMMIT;
SQL> SET QUIET COMMIT 'off';
SQL> COMMIT;
%SQL-F-NO_TXNOUT, No transaction outstanding
```

In the SQL module language or precompiler header, the clause QUIET COMMIT can be used to disable or enable error reporting for COMMIT and ROLLBACK when no transaction is active. The keyword ON or OFF must be used to enable or disable this feature. The following example enables QUIET COMMIT so that no error is reported if a COMMIT is executed when no transaction is active. For example:

```
MODULE TXN_CONTROL
LANGUAGE BASIC
PARAMETER COLONS
QUIET COMMIT ON

PROCEDURE S_TXN (SQLCODE);
SET TRANSACTION READ WRITE;

PROCEDURE C_TXN (SQLCODE);
COMMIT;
```

7.1.25.2 COMPOUND TRANSACTIONS Option

The SET COMPOUND TRANSACTIONS statement (for interactive and dynamic SQL) and the module header option COMPOUND TRANSACTIONS allows the programmer to control the SQL behavior for starting default transactions for compound statements.

By default, if there is no current transaction, SQL will start a transaction before executing a compound statement or stored procedure. However, this may conflict with the actions within the procedure, or may start a transaction for no reason if the procedure body does not perform any database access. This default is retained for backward compatibility for applications that may expect a transaction to be started for the procedure.

If COMPOUND TRANSACTIONS is set to EXTERNAL, then SQL starts a transaction before executing the procedure; otherwise, if it is set to INTERNAL, it allows the procedure to start a transaction as required by the procedure execution.

Examples

In interactive or dynamic SQL, the following SET command can be used to disable or enable transactions started by the SQL interface. The parameter to the SET command is a string literal or host variable containing the keyword INTERNAL or EXTERNAL. The keywords may be in any case (upper, lower, or mixed). For example:

```
SQL> SET COMPOUND TRANSACTIONS 'internal';
SQL> CALL START_TXN_AND_COMMIT ();
SQL> SET COMPOUND TRANSACTIONS 'external';
SQL> CALL UPDATE_EMPLOYEES (...);
```

In the SQL module language or precompiler header, the clause COMPOUND TRANSACTIONS can be used to disable or enable starting a transaction for procedures. The keyword INTERNAL or EXTERNAL must be used to enable or disable this feature.

```
MODULE TXN_CONTROL
LANGUAGE BASIC
PARAMETER COLONS
COMPOUND TRANSACTIONS INTERNAL

PROCEDURE S_TXN (SQLCODE);
BEGIN
SET TRANSACTION READ WRITE;
END;

PROCEDURE C_TXN (SQLCODE);
BEGIN
COMMIT;
END;
```

7.1.26 Undocumented Size Limit for Indexes with Keys Using Collating Sequences

Bug 586079

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

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

The space required for encoding the string must be taken into account when calculating the size of an index key against the limit of 255 bytes. Suppose a column defined with a collating sequence of GERMAN was used in an index. The length of that column is limited to a maximum of 225 characters because the key will be encoded in 254 bytes.

The following example demonstrates how a 233 character column, defined with a German collating sequence and included in an index, exceeds the index size limit of 255 bytes, even though the column is defined as less than 255 characters in length:

```
SQL> CREATE DATABASE
cont>      FILENAME 'TESTDB.RDB'
cont>      COLLATING SEQUENCE GERMAN GERMAN;
SQL> CREATE TABLE EMPLOYEE_INFO (
cont>      EMP_NAME CHAR (233));
SQL> CREATE INDEX EMP_NAME_IDX
cont>      ON EMPLOYEE_INFO (
cont>      EMP_NAME     ASC)
cont>      TYPE IS SORTED;
%RDB-E-NO_META_UPDATE, metadata update failed
-RDMS-F-INDTOOBIG, requested index is too big
```

7.1.27 Changes to RMU/REPLICATE AFTER/BUFFERS Command

The behavior of the RMU/REPLICATE AFTER/BUFFERS command has been changed. The /BUFFERS qualifier may be used with either the CONFIGURE option or the START option.

When using local buffers, the AIJ log roll−forward server (LRS) will use a minimum of 4096 buffers. The value provided to the /BUFFERS qualifier will be accepted, but it will be ignored if it is less than 4096. In addition, further parameters will be checked and the number of buffers may be increased if the resulting calculations are greater than the number of buffers specified by the /BUFFERS qualifier. If the database is configured to use more than 4096 AIJ request blocks (ARBs), then the number of buffers may be increased to the number of ARBs configured for the database. The LRS ensures that there are at least 10 buffers for every possible storage area in the database. Thus, if the total number of storage areas (both used and reserved) multiplied by 10 results in a greater number of buffers, that number will be used.

When global buffers are used, the number of buffers used by the AIJ log roll−forward server is determined as follows:

- If the /BUFFERS qualifier is omitted and the /ONLINE qualifier is specified, the number of buffers will default to the previously configured value, if any, or 256, whichever is larger.
- If the /BUFFERS qualifier is omitted and the /ONLINE qualifier is not specified or the /NOONLINE is specified, the number of buffers will default to the maximum number of global buffers allowed per user ("USER LIMIT"), or 256, whichever is larger.
- If the /BUFFERS qualifier is specified, that value must be at least 256, and it may not be greater than the maximum number of global buffers allowed per user ("USER LIMIT").

The /BUFFER qualifier now enforces a minimum of 256 buffers for the AIJ log roll−forward server. The maximum number of buffers allowed is still 524288 buffers.

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

Starting with Oracle Rdb V7.0.2.1, the RDMAIJ image has become a varianted image. Therefore, the information in section 2.12, "Step 10: Specify the Network Transport Protocol," of the *Oracle Rdb7 and Oracle CODASYL DBMS Guide to Hot Standby Databases* has become outdated in regards to setting up the RDMAIJSERVER object when using UCX as the network transport protocol. The UCX SET SERVICE command should now look similar to the following:

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

And for Oracle Rdb multiversion, it should look similar to the following:

```
$ UCX SET SERVICE RDMAIJ70 −
      /PORT=<port_number> −
      /USER_NAME=RDMAIJ70 −
      /PROCESS_NAME=RDMAIJ70 −
      /FILE=SYS$SYSTEM:RDMAIJSERVER70.com −
      /LIMIT=<limit>
```

The installation procedure for Oracle Rdb creates a user named RDMAIJ(nn) and places a file called RDMAIJSERVER(nn).com in SYS$SYSTEM and the RMONSTART(nn).COM command procedure will try to enable a service called RDMAIJ(nn) if UCX is installed and running.

Changing the RDMAIJ server to a multivarianted image does not impact installations using DECNet since the correct DECNet object is created during the Rdb installation.

7.1.29 CREATE INDEX Supported for Hot Standby

On page 1−13 of the *Guide to Hot Standby Databases*, the add new index operation is incorrectly listed as an offline operation not supported by Hot Standby. The CREATE INDEX operation is now fully supported by Hot Standby, as long as the transaction does not span all available AIJ journals, including emergency AIJ journals.

7.1.30 Dynamic OR Optimization Formats

Bug 711643

In Table C−2 on Page C−7 of the *Oracle Rdb7 Guide to Database Performance and Tuning*, the dynamic OR optimization format is incorrectly documented as [l:h...]n. The correct formats for Oracle Rdb Release 7.0 and later are [(l:h)n] and [l:h,l2:h2].

Chapter 8
Known Problems and Restrictions

This chapter describes problems, restrictions, and workarounds known to exist in Oracle Rdb Release 7.0.8.2.

8.1 Oracle Rdb Considerations

8.1.1 Some SQL92 Dialect−required Warnings Not Delivered

Bugs 3651847 and 4532451

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

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

  6.089385714285714E+004
1 row selected
%RDB−I−ELIM_NULL, null value eliminated in set function
SQL> show sqlca
```

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

Since there is a row in the SALARY_HISTORY table with a NULL in SALARY_AMOUNT, the set function AVG should report an informational message (and return a special warning level SQLSTATE/SQLCODE value).

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

8.1.2 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
```

8.1.3 RDMS−E−RTNSBC_INITERR, Cannot Init External Routine Server Site Executor

Execution of an external function (or procedure) with server site binding may unexpectedly fail.

The following message is an example of the error message you might see:

```
%RDB−E−EXTFUN_FAIL, external routine failed to compile or execute successfully
−RDMS−E−EXTABORT, routine NNNNNNNNN execution has been aborted
−RDMS−E−RTNSBC_INITERR, Cannot init. external routine server site executor;
reason XX
```

Where NNNNNNNNN is the function name and XX is a decimal value, e.g., 41.

While such errors are possible they are very unlikely to be seen, especially on systems that have had Oracle Rdb successfully installed. These errors usually indicate a problem with the environment. For instance, ensure that images RDMXSMvv.EXE, RDMXSRvv.EXE and RDMXSMPvv.EXE (where vv is the Rdb version) are installed and have the correct protections:

```
Directory DISK$:<SYS6.SYSCOMMON.SYSLIB>

RDMXSM70.EXE;3           183   8−APR−2004 09:37:31.36  (RWED,RWED,RWED,RE)
RDMXSMP70.EXE;3          159   8−APR−2004 09:37:31.54  (RWED,RWED,RWED,RE)
RDMXSR70.EXE;3            67   8−APR−2004 09:37:31.74  (RWED,RWED,RWED,RE)

Total of 3 files, 409 blocks.

DISK$:<SYS6.SYSCOMMON.SYSLIB>.EXE
   RDMXSM70;3       Open Hdr Shared            Lnkbl
DISK$:<SYS6.SYSCOMMON.SYSLIB>.EXE
```

```
   RDMXSMP70;3      Open Hdr Shared       Prot Lnkbl  Safe
DISK$:<SYS6.SYSCOMMON.SYSLIB>.EXE
   RDMXSR70;3       Open Hdr Shared            Lnkbl
```

8.1.4 AIJ Log Server Process May Loop Or Bugcheck

Bugs 2651475 and 1756433

Under unknown, but extremely rare conditions, on busy databases where the After Image Journal (AIJ) Log Server process is enabled, the ALS process has been observed to enter a loop condition writing AIJ information to the AIJ file(s).

In the worst case, this problem could cause all available journal files to be filled with repeating data. If no remedial action were to be taken, this condition could cause the database to be shutdown and the AIJ journals to be considered inaccessable.

The database is not corrupted by this problem.

Stopping and restarting the ALS process will clear the looping condition even if the ALS process must be stopped using the STOP/ID command.

Stopping the ALS process will not impact production as AIJ writes will automatically revert to the non−ALS behaviour.

In Oracle Rdb Release 7.0.7 the behaviour of Rdb has been changed so that should this problem be detected, the ALS process will automatically shutdown producing a bugcheck dump file. This will prevent any danger of filling all available journals and will ensure that the database remains available.

ALS may be safely restarted immediately as the conditions that cause such a loop are resolved during recovery of the ALS process.

8.1.5 Optimization of Check Constraints

Bug 1448422

When phrasing constraints using the "CHECK" syntax, a poorer strategy can be chosen by the optimizer than when the same or similar constraint is phrased using referential integrity (PRIMARY and FOREIGN KEY) constraints. Following is an example.

I have two tables T1 and T2, both with one column, and I wish to ensure that all values in table T1 exist in T2. Both tables have an index on the referenced field. I could use a PRIMARY KEY constraint on T2 and a FOREIGN KEY constraint on T1.

```
SQL> alter table t2
cont>   alter column f2 primary key not deferrable;
SQL> alter table t1
cont>   alter column f1 references t2 not deferrable;
```

When deleting from the PRIMARY KEY table, Oracle Rdb will only check for rows in the FOREIGN KEY table where the FOREIGN KEY has the deleted value. This can be seen as an index lookup on T1 in the retrieval strategy.

```
SQL> delete from t2 where f2=1;
Get     Temporary relation      Retrieval by index of relation T2
  Index name  I2 [1:1]
Index only retrieval of relation T1
  Index name  I1 [1:1]
%RDB-E-INTEG_FAIL, violation of constraint T1_FOREIGN1 caused operation to fail
```

The failure of the constraint is not important. What is important is that Rdb efficiently detects that only those rows in T1 with the same values as the deleted row in T2 can be affected.

It is necessary sometimes to define this type of relationship using CHECK constraints. This could be necessary because the presence of NULL values in the table T2 precludes the definition of a primary key on that table. This could be done with a CHECK constraint of the form:

```
SQL> alter table t1
cont>   alter column f1
cont>   check (f1 in (select * from t2)) not deferrable;
SQL> delete from t2 where f2=1;
Get     Temporary relation      Retrieval by index of relation T2
  Index name  I2 [1:1]
Cross block of 2 entries
  Cross block entry 1
    Index only retrieval of relation T1
      Index name  I1 [0:0]
  Cross block entry 2
    Conjunct        Aggregate-F1    Conjunct
    Index only retrieval of relation T2
      Index name  I2 [0:0]
%RDB-E-INTEG_FAIL, violation of constraint T1_CHECK1 caused operation to fail
```

The cross block is for the constraint evaluation. This retrieval strategy indicates that to evaluate the constraint, the entire index on table T1 is being scanned and for each key, the entire index in table T2 is being scanned.

The behavior can be improved somewhat by using an equality join condition in the select clause of the constraint:

```
SQL> alter table t1
cont>   alter column f1
cont>   check (f1 in (select * from t2 where f2=f1))
cont>      not deferrable;

or:

SQL> alter table t1
cont>   alter column f1
cont>   check (f1=(select * from t2 where f2=f1))
cont>      not deferrable;
```

In both cases, the retrieval strategy will look as follows:

```
SQL> delete from t2 where f2=1;
Get     Temporary relation      Retrieval by index of relation T2
```

```
  Index name  I2 [1:1]
Cross block of 2 entries
  Cross block entry 1
    Index only retrieval of relation T1
      Index name  I1 [0:0]
  Cross block entry 2
    Conjunct        Aggregate-F1    Conjunct
    Index only retrieval of relation T2
      Index name  I2 [1:1]
%RDB-E-INTEG_FAIL, violation of constraint T1_CHECK1 caused operation to fail
```

While the entire T1 index is scanned, at least the value from T1 is used to perform an index lookup on T2.

These restrictions result from semantic differences in the behavior of the "IN" and "EXISTS" operators with respect to null handling, and the complexity of dealing with non−equality join conditions.

To improve the performance of this type of integrity check on larger tables, it is possible to use a series of triggers to perform the constraint check. The following triggers perform a similar check to the constraints above.

```
SQL> create trigger t1_insert
cont>  after insert on t1
cont>  when (not exists (select * from t2 where f2=f1))
cont>    (error) for each row;
SQL> create trigger t1_update
cont>  after update on t1
cont>  when (not exists (select * from t2 where f2=f1))
cont>    (error) for each row;
SQL> ! A delete trigger is not needed on T1.
SQL> create trigger t2_delete
cont>  before delete on t2
cont>  when (exists (select * from t1 where f1=f2))
cont>    (error) for each row;
SQL> create trigger t2_modify
cont>  after update on t2
cont>  referencing old as t2o new as t2n
cont>  when (exists (select * from t1 where f1=t2o.f2))
cont>    (error) for each row;
SQL> ! An insert trigger is not needed on T2.
```

The strategy for a delete on T2 is now:

```
SQL> delete from t2 where f2=1;
Aggregate-F1    Index only retrieval of relation T1
  Index name  I1 [1:1]
Temporary relation      Get     Retrieval by index of relation T2
  Index name  I2 [1:1]
%RDB-E-TRIG_INV_UPD, invalid update; encountered error condition defined for
trigger
-RDMS-E-TRIG_ERROR, trigger T2_DELETE forced an error
```

The trigger strategy is the index only retrieval displayed first. You will note that the index on T1 is used to examine only those rows that may be affected by the delete.

Care must be taken when using this workaround as there are semantic differences in the operation of

the triggers, the use of "IN" and "EXISTS", and the use of referential integrity constraints.

This workaround is useful where the form of the constraint is more complex and cannot be phrased using referential integrity constraints. For example, if the application is such that the value in table T1 may be spaces or NULLs to indicate the absence of a value, the above triggers could easily be modified to allow for these semantics.

8.1.6 Dynamic Optimization Estimation Incorrect for Ranked Indices

The dynamic optimization process was incorrectly calculating the cost of scanning indices of type *SORTED RANKED*.

In the following example, the table being queried has the numbers one to one thousand in both fields. The different ranges used should result in a different estimated cost. However, in all cases the *ESTIM* phase computes the cost of scanning these indicies as 680:

```
SQL> select * from t where f1 between 1 and 2 and f2 between 2 and 1000;
~S#0003
Leaf#01 FFirst T Card=1000
  BgrNdx1 T1 [1:1] Fan=17
  BgrNdx2 T2 [1:1] Fan=17
~E#0003.01(1) Estim   Ndx:Lev/Seps/DBKeys 1:34/0\680 2:34/0\680
~E#0003.01(1) BgrNdx1 EofData  DBKeys=2  Fetches=2+0  RecsOut=1 #Bufs=1
~E#0003.01(1) FgrNdx  FFirst   DBKeys=1  Fetches=0+1  RecsOut=1`ABA
~E#0003.01(1) Fin     Buf      DBKeys=2  Fetches=0+0  RecsOut=1
          F1            F2
           2             2
1 row selected
SQL> select * from t where f1 between 2 and 1000 and f2 between 1 and 2;
~S#0004
Leaf#01 FFirst T Card=1000
  BgrNdx1 T1 [1:1] Fan=17
  BgrNdx2 T2 [1:1] Fan=17
~E#0004.01(1) Estim   Ndx:Lev/Seps/DBKeys 1:34/0\680 2:34/0\680
~E#0004.01(1) BgrNdx1 EofData  DBKeys=999  Fetches=0+10  RecsOut=1 #Bufs=10
~E#0004.01(1) FgrNdx  FFirst   DBKeys=1  Fetches=0+11  RecsOut=1`ABA
~E#0004.01(1) Fin     Buf      DBKeys=999  Fetches=0+0  RecsOut=1
          F1            F2
           2             2
1 row selected
```

In the first example (query 3), the index T1 on field F1 is the correct index to use, as the key range is very small. In the second example (query 4), the index T2 on field F2 is the correct index to use. However, in both cases, the indices are costed the same so no index reordering takes place.

Even in this small example, significantly more work is being performed in query 4 as can be observed from the I/O counts.

This is a known problem in Oracle Rdb and it will be fixed in a future release.

The only workaround is to use indices of *TYPE IS SORTED* rather than of *TYPE IS SORTED RANKED*.

8.1.7 Running Rdb Applications With the VMS Heap Analyzer

When trying to debug an Rdb application under the OpenVMS Heap Analyzer (by defining LIBRTL as SYS$LIBRARY:LIBRTL_INSTRUMENTED), the software will not attach to the database, and returns

```
RDB-E-UNAVAILABLE, Oracle Rdb is not available on your system
```

as if RDB is not running.

To solve this problem, there are two executables that must be installed as known images:

```
$install add sys$share:librtl_instrumented
$install add sys$share:dgit$libshr12
```

The error is misleading. Since parts of Rdb are installed as privileged images, any shareable images it references, AND any images they, in turn, reference, must also be 'known'. By redirecting LIBRTL to SYS$LIBRARY:LIBRTL_INSTRUMENTED, these extra images are referenced. If Rdb had directly referenced the new image, a more accurate error, such as:

```
%DCL-W-ACTIMAGE, error activating image xxxxx
```

would have been reported.

8.1.8 RMU/RECOVER/AREA Needs Area List

Bug 1778243

When doing an RMU/RECOVER/AREA, without specifying a list of area names, there will be a new version of the current active AIJ file created. This new version of the AIJ will have the next recovery sequence number. If a subsequent recovery is applied, an error is generated indicating that the original recovery sequence number cannot be found and the recovery will abort.

If a list of storage areas to be recovered is supplied, this behaviour does not occur and no new version of the journal is created. It is recommended as best practice to use a list of storage areas when recovering by area to avoid any subsequent confusion during recovery.

8.1.9 PAGE TRANSFER VIA MEMORY Disabled

Oracle internal testing has revealed that the "PAGE TRANSFER VIA MEMORY" option for global buffers is not as robust as is needed for the Mission Critical environments where Oracle Rdb is often deployed. This feature has been disabled in Oracle Rdb Version 7.0.xx. This feature is available in Rdb Version 7.1.

8.1.10 RMU/VERIFY Reports PGSPAMENT or PGSPMCLST Errors

RMU/VERIFY may sometimes report PGSPAMENT or PGSPMCLST errors when verifying storage areas. These errors indicate that the Space Area Management ("SPAM") page fullness threshold for a particular data page does not match the actual space usage on the data page. For a further discussion of SPAM pages, consult the Oracle Rdb Guide to Database Maintenance.

In general, these errors will not cause any adverse affect on the operation of the database. There is potential for space on the data page to not be totally utilized, or for a small amount of extra I/O to be expended when searching for space in which to store new rows. But unless there are many of these errors then the impact should be negligible.

It is possible for these inconsistencies to be introduced by errors in the Oracle Rdb product. When those cases are discovered, Oracle Rdb is corrected to prevent the introduction of the inconsistencies. It is also possible for these errors to be introduced during the normal operation of the product. The following scenario can leave the SPAM pages inconsistent:

1. A process inserts a row on a page, and updates the threshold entry on the corresponding SPAM page to reflect the new space utilization of the data page. The data page and SPAM pages are not flushed to disk.
2. Another process notifies the first process that it would like to access the SPAM page being held by the process. The first process flushes the SPAM page changes to disk and releases the page. Note that it has not flushed the data page.
3. The first process then terminates abnormally (for example, from the DCL STOP/IDENTIFICATION command). Since that process never flushed the data page to disk, it never wrote the changes to the Recovery Unit Journal (RUJ) file. Since there were no changes in the RUJ file for that data page, then the Database Recovery ("DBR") process did not need to rollback any changes to the page. The SPAM page retains the threshold update change made above even though the data page was never flushed to disk.

While it would be possible to create mechanisms to ensure that SPAM pages do not become out of synch with their corresponding data pages, the performance impact would not be trivial. Since these errors are relatively rare and the impact is not significant, the introduction of these errors is considered to be part of the normal operation of the Oracle Rdb product. If it can be proven that the errors are not due to the scenario above then Oracle Product Support should be contacted.

PGSPAMENT and PGSPMCLST errors may be corrected by doing any one of the following operations:

- Recreate the database by performing:
    1. SQL EXPORT
    2. SQL DROP DATABASE
    3. SQL IMPORT
- Recreate the database by performing:
    1. RMU/BACKUP
    2. SQL DROP DATABASE
    3. RMU/RESTORE
- Repair the SPAM pages by using the RMU/REPAIR command. Note that the RMU/REPAIR command does not write its changes to an after−image journal (AIJ) file. Therefore, Oracle recommends that a full database backup be performed immediately after using the RMU/REPAIR command.

8.1.11 Behavior Change in 'With System Logical_Name Translation' Clause

The way logical name translation is performed when 'with system logical_name translation' is specified in the 'location' clause of the 'create function' or the 'create routine' statements has changed. This change occured between OpenVMS VAX V5.5−2 and OpenVMS V7.1.

When 'with system logical_name translation' is specified, any logical name in the location string is expanded using only EXECUTIVE_MODE logical names. In OpenVMS VAX V5.5−2, the logical names are expanded from the SYSTEM logical name table only. In OpenVMS V7.1, the logical names are expanded from the first definition found when searching the logical name tables in (LNM$FILE_DEV) order.

Thus, if a logical is only defined in the EXECUTIVE_MODE SYSTEM table (and in no other EXECUTIVE_MODE tables), then there will be no apparent change in behavior. However, if a logical name has been defined in the EXECUTIVE_MODE GROUP table and in the EXECUTIVE_MODE SYSTEM table, then on OpenVMS VAX V5.5 the SYSTEM table translation will be used and on OpenVMS V7.1 the GROUP table translation will be used.

Oracle believes that this behavioral change is still in keeping with the secure intent of this clause for external routines. An OpenVMS user must have SYSNAM privilege to define an EXEC mode logical in any table. Therefore, it still provides a secure method of locating production sharable images for use by the Rdb server.

A future version of the Oracle Rdb SQL Reference manual will be reworded to remove the reference to the SYSTEM logical name table in the description. The keyword SECURE will be synonymous with SYSTEM in this context.

As an example, if the logical TEST_EXTRTN_1 is defined as:

```
$ show logical/access_mode=executive_mode test_extrtn_1
   "TEST_EXTRTN_1" = "NOSUCHIMG9" (LNM$PROCESS_TABLE)
   "TEST_EXTRTN_1" = "NOSUCHIMGA" (LNM$JOB_9D277AC0)
   "TEST_EXTRTN_1" = "NOSUCHIMGB" (TEST$GROUP_LOGICALS)
   "TEST_EXTRTN_1" = "DISK1:[TEST]EXTRTN.EXE" (LNM$SYSTEM_TABLE)
```

Then under OpenVMS VAX V5.5−2, TEST_EXTRTN_1 will be translated as "DISK1:[TEST]EXTRTN.EXE" whereas under OpenVMS V7.1 it will be translated as "NOSUCHIMG9".

8.1.12 Carry−Over Locks and NOWAIT Transactions Clarification

In NOWAIT transactions, the BLAST mechanism cannot be used. For the blocking user to receive the BLAST signal, the requesting user must request the locked resource with WAIT (which a NOWAIT transaction does not do). Oracle Rdb defines a resource called NOWAIT, which is used to indicate that a NOWAIT transaction has been started. When a NOWAIT transaction starts, the user requests the NOWAIT resource. All other database users hold a lock on the NOWAIT resource so that when the NOWAIT transaction starts, all other users are notified with a NOWAIT BLAST. The BLAST causes blocking users to release any carry−over locks. There can be a delay before the transactions with carry−over locks detect the presence of the NOWAIT transaction and release their carry−over locks. You can detect this condition by examining the stall messages. If the "Waiting for NOWAIT signal (CW)" stall message appears frequently,

then the application is probably experiencing a decrease in performance and you should consider disabling the carry−over lock behavior.

8.1.13 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.

8.1.14 Exclusive Access Transactions May Deadlock With RCS Process

If a table is frequently accessed by long running transactions that request READ/WRITE access reserving the table for EXCLUSIVE WRITE, and if the table has one or more indexes, you may experience deadlocks between the user process and the Row Cache Server (RCS) process.

There are at least three suggested workarounds to this problem:

1. Reserve the table for SHARED WRITE.
2. Close the database and disable row cache for the duration of the exclusive transaction.
3. Change the checkpoint interval for the RCS process to a time longer than the time required to complete the batch job and then trigger a checkpoint just before the batch job starts. Set the interval back to a smaller interval after the checkpoint completes.

8.1.15 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.

8.1.16 Clarification of the USER Impersonation Provided by the Oracle Rdb Server

Bug 551240

In Oracle Rdb V6.1, a new feature was introduced which allowed a user to attach (or connect) to a database by providing a username (USER keyword) and a password (USING keyword). This functionality allows the Rdb Server to impersonate those users in two environments.

- Remote Database Access. When DECnet is used as the remote transport, the Rdb/Dispatch layer of Oracle Rdb uses the provided username and password, or proxy access to create a remote process which matches the named user. However, in a remote connection over TCP/IP, the RDBSERVER process is always logged into RDB$REMOTE rather than a specified user account. In this case the Rdb Server impersonates the user by using the user's UIC (user identification code) during privilege checking. The UIC is assigned by the OpenVMS AUTHORIZE utility.
- SQL/Services database class services. When SQL/Services (possibly accessed by ODBC) accesses a database, it allows the user to logon to the database and the SQL/Services server then impersonates that user in the database.

When a database has access control established using OpenVMS rights identifiers, then access checking in these two environments does not work as expected. For example, if a user JONES was granted the rights identifier PAYROLL_ACCESS, then you would expect a table in the database with SELECT access granted to PAYROLL_ACCESS to be accessible to JONES. This does not currently work because the Rdb Server does not have the full OpenVMS security profile loaded, just the UIC. So only access granted to JONES is allowed.

This problem results in an error being reported such as the following from ODBC:

```
[Oracle][ODBC][Rdb]%RDB−E−NO_PRIV privileged by database facility (#−1028)
```

This is currently a restriction in this release of Oracle Rdb. In the Rdb V7.1 release, support is provided to inherit the users full security profile into the database.

8.1.17 Index STORE Clause WITH LIMIT OF Not Enforced in Single Partition Map

Bug 413410

An index which has a STORE clause with a single WITH LIMIT OF clause and no OTHERWISE clause doesn't validate the inserted values against the high limit. Normally values beyond the last WITH LIMIT OF clause are rejected during INSERT and UPDATE statements.

Consider this example:

```
create table PTABLE (
    NR
        INTEGER,
    A
        CHAR (2));
create index NR_IDX
    on PTABLE (
    NR)
    type is HASHED
    store using (NR)
        in EMPIDS_LOW
            with limit of (10);
```

When a value is inserted for NR that exceeds the value 10, then an error such as "%RDMS−E−EXCMAPLIMIT, exceeded limit on last partition in storage map for NR_IDX" should be generated. However, this error is only reported if the index has two or more partitions.

A workaround for this problem is to create a CHECK constraint on the column to restrict the upper limit. e.g. CHECK (NR <= 10). This check constraint should be defined as NOT DEFERRABLE and will be solved using an index lookup.

This problem will be corrected in a future version of Oracle Rdb.

8.1.18 Unexpected NO_META_UPDATE Error Generated by DROP MODULE ... CASCADE When Attached by PATHNAME

Bug 755182

The SQL statement DROP MODULE ... CASCADE may sometimes generate an unexpected NO_META_UPDATE error. This occurs when the session attaches to a database by PATHNAME.

```
SQL> drop module m1 cascade;
%RDB−E−NO_META_UPDATE, metadata update failed
−RDMS−F−OBJ_INUSE, object "M1P1" is referenced by M2.M2P1 (usage: Procedure)
−RDMS−E−MODNOTDEL, module "M1" has not been deleted
```

This error occurs because the CASCADE option is ignored because the Oracle CDD/Repository does not support CASCADE. The workaround is to attach by FILENAME and perform the metadata operation.

In a future version of Oracle Rdb, an informational message will be issued describing the downgrade from CASCADE to RESTRICT in such cases.

8.1.19 Application and Oracle Rdb Both Using SYS$HIBER

In application processes that use Oracle Rdb and the $HIBER system service (possibly via RTL routines such as LIB$WAIT), it is important that the application ensures that the event being waited for has actually occurred. Oracle Rdb uses $HIBER/$WAKE sequences for interprocess communications particularly when the ALS (AIJ Log Server) or the Row Cache features are enabled.

Oracle Rdb's use of the $WAKE system service can interfere with other users of $HIBER (such as the routine LIB$WAIT) that do not check for event completion, possibly causing a $HIBER to be unexpectedly resumed without waiting at all.

To avoid these situations, consider altering the application to use a code sequence that avoids continuing without a check for the operation (such as a delay or a timer firing) being complete.

The following pseudo−code shows one example of how a flag can be used to indicate that a timed−wait has completed correctly. The wait does not complete until the timer has actually fired and set TIMER_FLAG to TRUE. This code relies on ASTs being enabled.

```
ROUTINE TIMER_WAIT:
    BEGIN
    ! Clear the timer flag
    TIMER_FLAG = FALSE

    ! Schedule an AST for sometime in the future
    STAT = SYS$SETIMR (TIMADR = DELTATIME, ASTRTN = TIMER_AST)
    IF STAT <> SS$_NORMAL THEN LIB$SIGNAL (STAT)

    ! Hibernate.  When the $HIBER completes, check to make
    ! sure that TIMER_FLAG is set indicating that the wait
    ! has finished.
    WHILE TIMER_FLAG = FALSE
    DO  SYS$HIBER()
    END

ROUTINE TIMER_AST:
    BEGIN
    ! Set the flag indicating that the timer has expired
    TIMER_FLAG = TRUE

    ! Wake the main−line code
    STAT = SYS$WAKE ()
    IF STAT <> SS$_NORMAL THEN LIB$SIGNAL (STAT)
    END
```

Starting with OpenVMS V7.1, the LIB$WAIT routine has been enhanced via the FLAGS argument (with the LIB$K_NOWAKE flag set) to allow an alternate wait scheme (using the $SYNCH system service) that can avoid potential problems with multiple code sequences using the $HIBER system service. See the OpenVMS RTL Library (LIB$) Manual for more information about the LIB$WAIT routine.

8.1.20 IMPORT Unable to Import Some View Definitions

Bug 520651

View definitions that reference SQL functions, that is functions defined by the CREATE MODULE statement, cannot currently be imported by the SQL IMPORT statement. This is because the views are defined before the functions themselves exist.

The following example shows the errors from IMPORT.

```
IMPORTing view TVIEW
%SQL-F-NOVIERES, unable to import view TVIEW
%RDB-E-NO_META_UPDATE, metadata update failed
-RDB-E-OBSOLETE_METADA, request references metadata objects that no
longer exist
-RDMS-E-RTNNEXTS, routine FORMAT_OUT does not exist in this database
%RDB-E-OBSOLETE_METADA, request references metadata objects that no
longer exist
-RDMS-F-TABNOTDEF, relation TVIEW is not defined in database
```

The following script can be used to demonstrate the problem.

```
create database filename badimp;
create table t (sex char);

create module TFORMAT
    language SQL

    function FORMAT_OUT (:s char)
    returns char(4);
    return (case :s
            when 'F' then 'Female'
            when 'M' then 'Male'
            else NULL
            end);
end module;

create view TVIEW (m_f) as
    select FORMAT_OUT (sex) from t;

commit;

export database filename badimp into exp;
drop database filename badimp;
import database from exp filename badimp;
```

This restriction is lifted in the Rdb V7.1 releases. Currently the workaround is to save the view definitions and reapply them after the IMPORT completes.

This restriction does not apply to external functions, created using the CREATE FUNCTION statement, as these database objects are defined before tables and views.

8.1.21 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.

8.1.22 Lock Remastering and Hot Standby

When using the Hot Standby feature, Oracle recommends that the VMS distributed lock manager resource tree be mastered on the standby node where Hot Standby is started. This can be using any of the following methods:

- Disable dynamic lock remastering. This can be done dynamically by setting the SYSGEN parameter PE1 to the value 1.
  When using this option, be sure that Hot Standby is started on the node where the standby database is first opened.
- Increasing the LOCKDIRWT value for the LRS node higher than any other node in the same cluster. However, this is not a dynamic SYSGEN parameter, and a node re−boot is required.

Failure to prevent dynamic lock remastering may cause severe performance degradation for the standby database, which ultimately may be reflected by decreased master database transaction throughput.

8.1.23 RDB_SETUP Privilege Error

Rdb Web Agent V3.0 exposes a privilege problem with Rdb V7.0 and later. This will be fixed in the next Rdb 7.0 release.

The RDB_SETUP function fails with %RDB−E−NO_PRIV, privilege denied by database facility.

It appears that the only workaround is to give users DBADM privilege. Oracle Corporation does not recommend giving users the DBADM privilege.

8.1.24 Starting Hot Standby on Restored Standby Database May Corrupt Database

If a standby database is modified outside of Hot Standby, then backed up and restored, Hot Standby will appear to start up successfully but will corrupt the standby database. A subsequent query of the database will return unpredictable results, possibly in a bugcheck in DIOFETCH$FETCH_ONE_LINE. When the standby database is restored from a backup of itself, the database is marked as unmodified. Therefore, Hot Standby cannot tell whether the database had been modified before the backup was taken.

WORKAROUND: None.

8.1.25 Restriction on Compound Statement Nesting Levels

The use of multiple nesting levels of compound statements such as CASE or IF−THEN−ELSE within multistatement procedures can result in excessive memory usage during the compile of the procedure. Virtual memory problems have been reported with 10 or 11 levels of nesting. The following example shows an outline of the type of nesting that can lead to this problem.

```
CREATE MODULE MY_MOD LANGUAGE SQL
PROCEDURE MY PROCEDURE
    ( PARAMETERS .....);

BEGIN
  DECLARE ....;

SET :VARS = 0;

SELECT .....;
GET DIAGNOSTICS EXCEPTION 1 :FLAG = RETURNED_SQLCODE;
CASE :FLAG                        ! Case #1
       WHEN 100 THEN SET ...;
       WHEN −811 THEN SET ...;
       WHEN 0 THEN
         SET ...; SELECT ...;
         GET DIAGNOSTICS EXCEPTION 1 :FLAG = RETURNED_SQLCODE;
         CASE :FLAG            ! Case #2
            WHEN 0 THEN SET ...;
            WHEN −811 THEN SET ...;
            WHEN 100 THEN
               UPDATE...; SET ...;
               GET DIAGNOSTICS EXCEPTION 1 :FLAG = RETURNED_SQLCODE;
               IF :FLAG= 100 THEN SET ...;              ! #1
               ELSE
                IF :FLAG < 0 THEN SET...;               ! #2
               ELSE
                   DELETE  ...
                   GET DIAGNOSTICS EXCEPTION 1 :FLAG = RETURNED_SQLCODE;
                   IF :FLAG= 100 THEN SET...;           ! #3
                      SET ...;
                   ELSE
                    IF :FLAG < 0 THEN SET...;           ! #4
                    ELSE
                     IF IN_CHAR_PARAM = 'S' THEN        ! #5
                      UPDATE ...
                      GET DIAGNOSTICS EXCEPTION 1 :FLAG = RETURNED_SQLCODE;
                        IF :FLAG= 100 THEN SET ...;     ! #6
```

```
                        ELSE
                         IF :FLAG < 0 THEN SET...;      ! #7
                         END IF;                        ! #7
                       END IF;                         ! #6
                     END IF;                           ! #5

                     IF :FLAG = 0 THEN                 ! #5
                       UPDATE ...
                       GET DIAGNOSTICS EXCEPTION 1 :FLAG = RETURNED_SQLCODE;
                       IF :FLAG= 100 THEN SET ...;     ! #6
                       ELSE
                          IF :FLAG < 0 THEN SET ...;   ! #7
                          ELSE
                             DELETE ...
                             GET DIAGNOSTICS EXCEPTION 1 :FLAG = RETURNED_SQLCODE:
                             IF :FLAG= 100 THEN SET ...;       ! #8
                             ELSE
                               IF :FLAG < 0 THEN SET ...;      ! #9
                               ELSE
                                 DELETE ...;
                                 GET DIAGNOSTICS EXCEPTION 1 :FLAG = RETURNED_SQLCODE;
                                 IF :FLAG= 100 THEN SET ...;   ! #10
                                    SET ...;
                                 ELSE
                                    IF :FLAG < 0 THEN SET ...;  ! #11
                                    END IF; (11 end if's  for #11 − #1)
            ELSE SET ...;
            END CASE;                ! Case #2
       ELSE SET ...;
       END CASE;                     ! Case #1
END;
END MODULE;
```

Workaround: Reduce the complexity of the multistatement procedure. Use fewer levels of compound statement nesting by breaking the multistatement procedure into smaller procedures or by using the CALL statement to execute nested stored procedures.

8.1.26 Back Up All AIJ Journals Before Performing a Hot Standby Switchover Operation

Prior to performing a proper Hot Standby switchover operation from the old master database to the new master database (old standby database), be sure to back up ALL AIJ journals.

If you do not back up the AIJ journals on the old master database prior to switchover, they will be initialized by the Hot Standby startup operation, and you will not have a backup of those AIJ journals.

Failure to back up these journals may place your new master database at risk of not being able to be recovered, requiring another fail−over in the event of system failure.

8.1.27 Concurrent DDL and Read−Only Transaction on the Same Table Not Compatible

It is possible that a read−only transaction could generate a bugcheck at DIOBND$FETCH_AIP_ENT + 1C4 if there is an active, uncommitted transaction that is making metadata changes to the same table. Analysis shows

that the snapshot transaction is picking up stale metadata information. Depending on what metatdata modifications are taking place, it is possible for metadata information to be removed from the system tables but still exist in the snapshot file. When the read−only transaction tries to use that information, it no longer exists and causes a bugcheck.

The following example shows the actions of the two transactions:

```
A:                              B:
attach
set transaction read write
                                attach
                                set transaction read only
drop index emp_last_name
                                select * from employees
                                ...bugcheck...
```

The only workaround is to avoid running the two transactions together.

8.1.28 Oracle Rdb and the SRM_CHECK Tool

The Alpha Architecture Reference Manual, Third Edition (AARM) describes strict rules for using interlocked memory instructions. The HP Alpha 21264 (EV6) processor and all future Alpha processors are more stringent than their predecessors in their requirement that these rules be followed. As a result, code that has worked in the past despite noncompliance may now fail when executed on systems featuring the new 21264 processor.

Oracle Rdb Release 7.0.3 supports the HP Alpha 21264 (EV6) processor. Oracle has performed extensive testing and analysis of the Rdb code to ensure that it is compliant with the rules for using interlocked memory instructions.

However, customers using the HP supplied SRM_CHECK tool may find that several of the Oracle Rdb images cause the tool to report potential alpha architecture violations. Although SRM_CHECK can normally identify a code section in an image by the section's attributes, it is possible for OpenVMS images to contain data sections with those same attributes. As a result, SRM_CHECK may scan data as if it were code, and occasionally, a block of data may look like a noncompliant code sequence. This is the case with the Oracle Rdb supplied images. There is no actual instruction stream violation.

However, customers must use the SRM_CHECK tool on their own application executable image files. It is possible that applications linked with very old version of Oracle Rdb (versions prior to Oracle Rdb Release 6.0−05) could have included illegal interlocked memory instruction sequences produced by very old versions of compilers. This code was included in the Oracle Rdb object library files for some very old versions of Oracle Rdb.

If errant instruction sequences are detected in the objects supplied by the Oracle Rdb object libraries, the correct action is to relink the application with a more−current version of Oracle Rdb.

Additional information about the HP Alpha 21264 (EV6) processor interlocked memory instructions issues is available at:

http://www.openvms.digital.com/openvms/21264_considerations.html

8.1.29 Oracle RMU Checksum_Verification Qualifier

The Oracle Rdb RMU BACKUP database backup command includes a Checksum_Verification qualifier.

Specifying Checksum_Verification requests that the RMU Backup command verify the checksum stored on each database page before it is backed up, thereby providing end−to−end error detection on the database I/O.

The Checksum_Verification qualifier uses additional CPU resources but can provide an extra measure of confidence in the quality of the data backed up. Use of the Checksum_Verification qualifier offers an additional level of data security and use of the Checksum_Verification qualifier permits Oracle RMU to detect the possibility that the data it is reading from these disks has only been partially updated.

Note, however, that if you specify the Nochecksum_Verification qualifier, and undetected corruptions exist in your database, the corruptions are included in your backup file and restored when you restore the backup file. Such a corruption might be difficult to recover from, especially if it is not detected until weeks or months after the restore operation is performed.

Oracle Corporation recommends that you use the Checksum_Verification qualifier with all database backup operations because of the improved data integrity this qualifier provides.

Unfortunately, due to an oversight, for versions of Oracle Rdb prior to Version 8.0, the default for online backups is the Nochecksum_Verification qualifier. When you do not specify the Checksum_Verification qualifier on all of your RMU database backup commands.

8.1.30 Do Not Use HYPERSORT with RMU/OPTIMIZE/AFTER_JOURNAL (Alpha)

OpenVMS Alpha V7.1 introduced the high−performance Sort/Merge utility (also known as HYPERSORT). This utility takes advantage of the Alpha architecture to provide better performance for most sort and merge operations.

The high−performance Sort/Merge utility supports a subset of the SOR routines. Unfortunately, the high−performance Sort/Merge utility does not support several of the interfaces used by the RMU/OPTIMIZE/AFTER_JOURNAL command. In addition, the high−performance Sort/Merge utility reports no error or warning when being called with the unsupported options used by the RMU/OPTIMIZE/AFTER_JOURNAL command.

For this reason, the use of the high−performance Sort/Merge utility is not supported for the RMU/OPTIMIZE/AFTER_JOURNAL command. Do not define the logical name SORTSHR to reference HYPERSORT.EXE.

8.1.31 Restriction on Using /NOONLINE with Hot Standby

When a user process is performing a read−only transaction on a standby database, an attempt to start replication on the standby database with the /NOONLINE qualifier will fail with the following error, and the database will be closed cluster−wide:

```
%RDMS−F−OPERCLOSE, database operator requested database shutdown
```

In a previous release, the following error was returned and the process doing the read−only transaction was not affected:

```
%RDMS−F−STBYDBINUSE, standby database cannot be exclusively accessed for
replication
```

As a workaround, if exclusive access is necessary to the standby database, terminate any user processes before starting replication with the /NOONLINE qualifier.

This restriction is due to another bug fix and will be lifted in a future release of Oracle Rdb.

8.1.32 SELECT Query May Bugcheck with PSII2SCANGETNEXTBBCDUPLICATE Error

Bug 683916

A bugcheck could occur when a ranked B−tree index is used in a query after a database has been upgraded to Release 7.0.1.3. This is a result of index corruption that was introduced in previous versions of Oracle Rdb. This corruption has been fixed and indexes created using Release 7.0.1.3 will not be impacted.

As a workaround, delete the affected index and re−create it under Oracle Rdb Release 7.0.1.3 or later.

8.1.33 DBAPack for Windows 3.1 is Deprecated

Oracle Enterprise Manager DBAPack will no longer be supported for use on Windows 3.1.

8.1.34 Determining Mode for SQL Non−Stored Procedures

Bug 506464.

Although stored procedures allow parameters to be defined with the modes IN, OUT, and INOUT, there is no similar mechanism provided for SQL module language or SQL precompiled procedures. However, SQL still associates a mode with a parameter using the following rules:

Any parameter which is the target of an assignment is considered an OUT parameter. Assignments consist of the following:

- The parameter is assigned a value with the SET or GET DIAGNOSTICS statement. For example:

```
set :p1 = 0;
get diagnostics :p2 = TRANSACTION_ACTIVE;
```

- The parameter is assigned a value with the INTO clause of an INSERT, UPDATE, or SELECT statement. For example:

```
insert into T (col1, col2)
    values (...)
    returning dbkey into :p1;

update accounts
    set account_balance = account_balance + :amount
    where account_number = :p1
```

```
    returning account_balance
    into :current_balance;

select last_name
    into :p1
    from employees
    where employee_id = '00164';
```

- The parameter is passed on a CALL statement as an OUT or INOUT argument. For example:

```
begin
call GET_CURRENT_BALANCE (:p1);
end;
```

Any parameter that is the source for a query is considered an IN parameter. Query references include:

- The parameter appears in the SELECT list, WHERE or HAVING clauses of a SELECT, or DELETE statement. For example:

```
select :p1 || last_name, count(*)
    from T
    where last_name like 'Smith%'
    group by last_name
    having count(*) > :p2;

delete from T
    where posting_date < :p1;
```

- The parameter appears on the right side of the assignment in a SET statement or SET clause of an UPDATE statement. For example:

```
set :p1 = (select avg(salary)
            from T
            where department = :p2);
update T
    set col1 = :p1
    where ...;
```

- The parameter is used to provide a value to a column in an INSERT statement. For example:

```
insert into T (col1, col2)
    values (:p1, :p2);
```

- The parameter is referenced by an expression in a TRACE, CASE, IF/ELSEIF, WHILE statement, or by the DEFAULT clause of a variable declaration. For example:

```
begin
declare :v integer default :p1;
DO_LOOP:
while :p2 > :p1
loop
    if :p1 is null then
        leave DO_LOOP;
    end if;
    set :p2 = :p2 + 1;
    ...;
    trace 'Loop at ', :p2;
end loop;
end;
```

- The parameter is passed on a CALL statement as an INOUT or IN argument. For example:

```
begin
```

```
    call SET_LINE_SPEED (:p1);
    end;
```

SQL only copies values from the client (application parameters) to the procedure running in the database server if it is marked as either an IN or INOUT parameter. SQL only returns values from the server to the client application parameter variables if the parameter is an OUT or INOUT parameter.

If a parameter is considered an OUT only parameter, then it must be assigned a value within the procedure, otherwise the result returned to the application is considered undefined. This could occur if the parameter is used within a conditional statement such as CASE or IF/ELSEIF. In the following example, the value returned by :p2 would be undefined if :p1 were negative or zero:

```
begin
if :p1 > 0 then
    set :p2 = (select count(*)
                from T
                where col1 = :p1);
end if;
end;
```

It is the responsibility of the application programmer to ensure that the parameter is correctly assigned values within the procedure. A workaround is to either explicitly initialize the OUT parameter, or make it an INOUT parameter. For example:

```
begin
if :p1 > 0 then
    set :p2 = (select count(*)
                from T
                where col1 = :p1);
elseif :p2 is null then
    begin
    end;
end if;
end;
```

The empty statement will include a reference to the parameter to make it an IN parameter as well as an OUT parameter.

8.1.35 DROP TABLE CASCADE Results in %RDB−E−NO_META_UPDATE Error

An error could result when a DROP TABLE CASCADE statement is issued. This occurs when the following conditions apply:

- A table is created with an index defined on the table.
- A storage map is created with a placement via index.
- The storage map is a vertical record partition storage map with two or more STORE COLUMNS clauses.

The error message given is %RDB−E−NO_META_UPDATE, metadata update failed.

The following example shows a table, index, and storage map definition followed by a DROP TABLE CASCADE statement and the resulting error message:

```
SQL> CREATE TABLE VRP_TABLE ( ID INT, ID2 INT);
SQL> COMMIT;
SQL> CREATE UNIQUE INDEX VRP_IDX ON VRP_TABLE (ID)
SQL> STORE IN EMPIDS_LOW;
SQL> COMMIT;
SQL> CREATE STORAGE MAP VRP_MAP
cont> FOR VRP_TABLE
cont> PLACEMENT VIA INDEX VRP_IDX
cont> ENABLE COMPRESSION
cont> STORE COLUMNS (ID)
cont> IN EMPIDS_LOW
cont> STORE COLUMNS (ID2)
cont> IN EMPIDS_MID;
SQL> COMMIT;
SQL>
SQL> DROP TABLE VRP_TABLE CASCADE;
SQL> -- Index VRP_IDX is also being dropped.
%RDB-E-NO_META_UPDATE, metadata update failed
-RDMS-E-WISH_LIST, feature not implemented yet
-RDMS-E-VRPINVALID, invalid operation for storage map "VRP_MAP"
```

The workaround to this problem is to first delete the storage map, and then delete the table using the CASCADE option. The following example shows the workaround. The SHOW statement indicates that the table, index, and storage map were deleted:

```
SQL> DROP STORAGE MAP VRP_MAP;
SQL> DROP TABLE VRP_TABLE CASCADE;
SQL> -- Index VRP_IDX is also being dropped.
SQL> COMMIT;
SQL> SHOW TABLE VRP_TABLE
No tables found
SQL> SHOW INDEX VRP_IDX
No indexes found
SQL> SHOW STORAGE MAP VRP_MAP
No Storage Maps Found
```

This problem will be corrected in a future version of Oracle Rdb.

8.1.36 Bugcheck Dump Files with Exceptions at COSI_CHF_SIGNAL

In certain situations, Oracle Rdb bugcheck dump files will indicate an exception at COSI_CHF_SIGNAL. This location is, however, not the address of the actual exception. The actual exception occurred at the previous call frame on the stack (the one listed as the next "Saved PC" after the exception).

For example, consider the following bugcheck file stack information:

```
$ SEARCH RDSBUGCHK.DMP "EXCEPTION","SAVED PC","-F-","-E-"

***** Exception at 00EFA828 : COSI_CHF_SIGNAL + 00000140
%COSI-F-BUGCHECK, internal consistency failure
Saved PC = 00C386F0 : PSIINDEX2JOINSCR + 00000318
Saved PC = 00C0BE6C : PSII2BALANCE + 0000105C
Saved PC = 00C0F4D4 : PSII2INSERTT + 000005CC
Saved PC = 00C10640 : PSII2INSERTTREE + 000001A0
    .
    .
```

.

In this example, the exception actually occurred at PSIINDEX2JOINSCR offset 00000318. If you have a bugcheck dump with an exception at COSI_CHF_SIGNAL, it is important to note the next "Saved PC" because it will be needed when working with Oracle Rdb Support Services.

8.1.37 Interruptions Possible when Using Multistatement or Stored Procedures

Long running multistatement or stored procedures can cause other users in the database to be interrupted by holding resources needed by those other users. Some resources obtained by the execution of a multistatement or stored procedure will not be released until the multistatement or stored procedure finishes. This problem can be encountered even if the statement contains COMMIT or ROLLBACK statements.

The following example demonstrates the problem. The first session enters an endless loop; the second session attempts to backup the database, but it is permanently interrupted:

Session 1

```
SQL> ATTACH 'FILE MF_PERSONNEL';
SQL> CREATE FUNCTION LIB$WAIT (IN REAL BY REFERENCE)
cont> RETURNS INT;
cont> EXTERNAL NAME LIB$WAIT
cont> LOCATION 'SYS$SHARE:LIBRTL.EXE'
cont> LANGUAGE GENERAL
cont> GENERAL PARAMETER STYLE
cont> VARIANT;
SQL> COMMIT;
SQL> EXIT;

$ SQL
SQL> ATTACH 'FILE MF_PERSONNEL';
SQL> BEGIN
cont> DECLARE :LAST_NAME LAST_NAME_DOM;
cont> DECLARE :WAIT_STATUS INTEGER;
cont> LOOP
cont> SELECT LAST_NAME INTO :LAST_NAME
cont>   FROM EMPLOYEES WHERE EMPLOYEE_ID = '00164';
cont> ROLLBACK;
cont> SET :WAIT_STATUS = LIB$WAIT (5.0);
cont> SET TRANSACTION READ ONLY;
cont> END LOOP;
cont> END;
```

Session 2

```
$ RMU/BACKUP/LOG/ONLINE MF_PERSONNEL MF_PERSONNEL
```

From a third session we can see that the backup process is waiting for a lock held in the first session:

```
$ RMU/SHOW LOCKS /MODE=BLOCKING MF_PERSONNEL
================================================================================
SHOW LOCKS/BLOCKING Information
================================================================================
```

```
-------------------------------------------------------------------------------
Resource: nowait signal

          ProcessID Process Name        Lock ID   System ID Requested Granted
          --------- ---------------     --------- --------- --------- -------
Waiting:  20204383  RMU BACKUP.....     5600A476  00010001  CW        NL
Blocker:  2020437B  SQL............     3B00A35C  00010001  PR        PR
$
```

There is no workaround for this restriction. When the multistatement or stored procedure finishes execution, the resources needed by other processes will be released.

8.1.38 Row Cache Not Allowed on Standby Database While Hot Standby Replication Is Active

The row cache feature may not be active on a Hot Standby database while replication is taking place. The Hot Standby feature will not start if row cache is active on the standby database.

This restriction exists because rows in the row cache are accessed using logical dbkeys. However, information transferred to the Hot Standby database from the after−image journal facility only contains physical dbkeys. Because there is no way to maintain rows in the cache using the Hot Standby processing, the row cache must be disabled on the standby database when the standby database is open and replication is active. The master database is not affected; the row cache feature and the Hot Standby feature may be used together on a master database.

The row cache feature should be identically configured on the master and standby databases in the event failover occurs, but the row cache feature must not be activated on the standby database until it becomes the master.

A new command qualifier, ROW_CACHE=DISABLED, has been added to the RMU/OPEN command to disable the row cache feature on the standby database. To open the Hot Standby database prior to starting replication, use the ROW_CACHE=DISABLED qualifier on the RMU/OPEN command.

8.1.39 Hot Standby Replication Waits when Starting if Read−Only Transactions Running

Hot Standby replication will wait to start if there are read−only (snapshot) transactions running on the standby database. The log roll−forward server (LRS) will wait until the read−only transactions commit, and then replication will continue.

This is an existing restriction of the Hot Standby software. This release note is intended to complement the Hot Standby documentation.

8.1.40 Error when Using the SYS$LIBRARY:SQL_FUNCTIONS70.SQL Oracle Functions Script

If your programming environment is not set up correctly, you may encounter problems running the

SYS$LIBRARY:SQL_FUNCTIONS70.SQL script used to set up the Oracle7 functions being supplied with Oracle Rdb.

The following example shows the error:

```
%RDB-E-EXTFUN_FAIL, external routine failed to compile or execute successfully
-RDMS-E-INVRTNUSE, routine RDB$ORACLE_SQLFUNC_INTRO can not be used, image
"SQL$FUNCTIONS" not activated
-RDMS-I-TEXT, Error activating image
DISK:[DIR]SQL$FUNCTIONS.;, File not found
```

To resolve this problem, use the @SYS$LIBRARY:RDB$SETVER to set up the appropriate logical names. This will be necessary for programs that use the functions as well.

In a standard environment, use the command shown in the following example:

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

In a multiversion environment, use the command shown in the following example:

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

8.1.41 DEC C and Use of the /STANDARD Switch

Bug 394451

The SQL$PRE compiler examines the system to know which dialect of C to generate. That default can be overwritten by using the /CC=[DECC/VAXC] switch. The /STANDARD switch should not be used to choose the dialect of C.

Support for DEC C was added to the product with V6.0 and this note is meant to clarify that support, not to indicate a change. It is possible to use /STANDARD=RELAXED_ANSI89 or /STANDARD=VAXC correctly, but this is not recommended.

The following example shows both the right and wrong way to compile an Oracle Rdb SQL program. Assume a symbol SQL$PRE has been defined, and DEC C is the default C compiler on the system:

```
 $ SQL$PRE/CC  ! This is correct.
 $ SQL$PRE/CC=DECC ! This is correct.
 $ SQL$PRE/CC=VAXC ! This is correct.

 $ SQL$PRE/CC/STANDARD=VAXC   ! This is incorrect.
```

Notice that the /STANDARD switch has other options in addition to RELAXED_ANSI89 and VAX C. Those are also not supported.

8.1.42 Excessive Process Page Faults and Other Performance Considerations During Oracle Rdb Sorts

Excessive hard or soft page faulting can be a limiting factor of process performance. Sometimes this page faulting occurs during Oracle Rdb sort operations. This note describes how page faulting can occur and some ways to help control, or at least understand, it.

One factor contributing to Oracle Rdb process page faulting is sorting operations. Common causes of sorts include the SQL GROUP BY, ORDER BY, UNION, and DISTINCT clauses specified for query and index creation operations. Defining the logical name RDMS$DEBUG_FLAGS to "RS" can help determine when Oracle Rdb sort operations are occurring and to display the sort keys and statistics.

Oracle Rdb includes its own copy of the OpenVMS SORT32 code within the Oracle Rdb images and does not generally call the routines in the OpenVMS run−time library. A copy of the SORT32 code is used to provide stability between versions of Oracle Rdb and OpenVMS and because Oracle Rdb calls the sort routines from executive processor mode which is difficult to do using the SORT32 sharable image. Database import and RMU load operations call the OpenVMS sort run−time library.

At the beginning of a sort operation, the sort code allocates some memory for working space. The sort code uses this space for buffers, in−memory copies of the data, and sorting trees.

Sort code does not directly consider the process quotas or parameters when allocating memory. The effects of WSQUOTA and WSEXTENT are indirect. At the beginning of each sort operation, the sort code attempts to adjust the process' working set to the maximum possible size using the $ADJWSL system service specifying a requested working set limit of %X7FFFFFFF pages (the maximum possible). Sort code then uses a value of 75% of the returned working set for virtual memory scratch space. The scratch space is then initialized and the sort begins.

The initialization of the scratch space generally causes page faults to access the pages newly added to the working set. Pages that were in the working set already may be faulted out as new pages are faulted in. Once the sort operation completes, the pages that may have been faulted out of the working set are likely to be faulted back into the working set.

When a process' working set is limited by the working set quota (WSQUOTA) parameter and the working set extent (WSEXTENT) parameter is a much larger value, the first call to the sort routines can cause many page faults as the working set grows. Using a value of WSEXTENT that is closer to WSQUOTA can help reduce the impact of this case.

With some OpenVMS versions, AUTOGEN sets the SYSGEN parameter PQL_MWSEXTENT equal to the WSMAX parameter. This means that all processes on the system end up with WSEXTENT the same as WSMAX. Because WSMAX might be quite high, sorting might result in excessive page faulting. You may want to explicitly set PQL_MWSEXTENT to a lower value if this is the case on your system.

Sort work files are another factor to consider when tuning Oracle Rdb sort operations. When the operation cannot be done in available memory, sort code will use temporary disk files to hold the data as it is being sorted. The *Oracle Rdb Guide to Performance and Tuning* contains more detailed information about sort work files.

The logical name RDMS$BIND_SORT_WORKFILES specifies how many work files sort code is to use if work files are required. The default is 2, and the maximum number is 10. The work files can be individually

controlled by the SORTWORKn logical names (where n is from 0 through 9). You can increase the efficiency of sort operations by assigning the location of the temporary sort work files to different disks. These assignments are made by using up to 10 logical names, SORTWORK0 through SORTWORK9.

Normally, sort code places work files in the user's SYS$SCRATCH directory. By default, SYS$SCRATCH is the same device and directory as the SYS$LOGIN location. Spreading the I/O load over many disks improves efficiency as well as performance by taking advantage of the system resources and helps prevent disk I/O bottlenecks. Specifying that a user's work files will reside on separate disks permits overlap of the sort read/write cycle. You may also encounter cases where insufficient space exists on the SYS$SCRATCH disk device, such as when Oracle Rdb builds indexes for a very large table. Using the SORTWORK0 through SORTWORK9 logical names can help you avoid this problem.

Note that sort code uses the work files for different sorted runs, and then merges the sorted runs into larger groups. If the source data is mostly sorted, then not every sort work file may need to be accessed. This is a possible source of confusion because even with 10 sort work files, it is possible to exceed the capacity of the first sort file, and the sort operation will fail never having accessed the remaining 9 sort work files.

Note that the logical names RDMS$BIND_WORK_VM and RDMS$BIND_WORK_FILE do not affect or control the operation of sort. These logical names are used to control other temporary space allocations within Oracle Rdb.

8.1.43 Performance Monitor Column Mislabeled

The File IO Overview statistics screen, in the Rdb Performance Monitor, contains a column labeled Pages Checked. The column should be labeled Pages Discarded to correctly reflect the statistic displayed.

8.1.44 Restriction Using Backup Files Created Later than Oracle Rdb Release 7.0.1

Bug 521583

Backup files created using Oracle Rdb releases later than 7.0.1 cannot be restored using Oracle Rdb Release 7.0.1. To fix a problem in a previous release, some internal backup file data structures were changed. These changes are not backward compatible with Oracle Rdb Release 7.0.1.

If you restore the database using such a backup file, then any attempt to access the restored database may result in unpredictable behavior, even though a verify operation may indicate no problems.

There is no workaround to this problem. For this reason, Oracle Corporation strongly recommends performing a full and complete backup both before and after the upgrade from Release 7.0.1 to later releases of Oracle Rdb.

8.1.45 RMU Backup Operations and Tape Drive Types

When using more than one tape drive for an RMU backup operation, all the tape drives must be of the same type. For example, all the tape drives must be either TA90s or TZ87s or TK50s. Using different tape drive types (one TK50 and one TA90) for a single database backup operation may make database restoration difficult or impossible.

Oracle RMU attempts to prevent using different tape drive densities during a backup operation, but is not able to detect all invalid cases and expects that all tape drives for a backup are of the same type.

As long as all the tapes used during a backup operation can be read by the same type of tape drive during a restore operation, the backup is likely to be valid. This may be the case, for example, when using a TA90 and a TA90E.

Oracle recommends that, on a regular basis, you test your backup and recovery procedures and environment using a test system. You should restore the databases and then recover them using AIJs to simulate failure recovery of the production system.

Consult the *Oracle Rdb Guide to Database Maintenance*, the *Oracle Rdb Guide to Database Design and Definition*, and the *Oracle RMU Reference Manual* for additional information about Oracle Rdb backup and restore operations.

8.1.46 Use of Oracle Rdb from Shared Images

Bug 470946

If code in the image initialization routine of a shared image makes any calls into Oracle Rdb, through SQL or any other means, access violations or other unexpected behavior may occur if Oracle Rdb's images have not had a chance to do their own initialization.

To avoid this problem, applications must do one of the following:

- Do not make Oracle Rdb calls from the initialization routines of shared images.
- Link in such a way that the RDBSHR.EXE image initializes first. This can be done by placing the reference to RDBSHR.EXE and any other Oracle Rdb shared images last in the linker options file.

8.1.47 Restriction Added for CREATE STORAGE MAP on Table with Data

Oracle Rdb Releease 7.0 added support that allows a storage map to be added to an existing table which contains data. The restrictions listed for Oracle Rdb Release 7.0 were:

- The storage map must be a simple map that references only the default storage area and represents the current (default) mapping for the table. The default storage area is either RDB$SYSTEM or the area name provided by the CREATE DATABASE...DEFAULT STORAGE AREA clause.
- The new map cannot change THRESHOLDS or COMPRESSION for the table, nor can it use the PLACEMENT VIA INDEX clause. It can only contain one area and cannot be vertically partitioned. This new map simply describes the mapping as it exists by default for the table.

This release of Rdb adds the additional restriction that the storage map may not include a WITH LIMIT clause for the storage area. The following example shows the reported error:

```
SQL> CREATE TABLE MAP_TEST1 (A INTEGER, B CHAR(10));
SQL> CREATE INDEX MAP_TEST1_INDEX ON MAP_TEST1 (A);
SQL> INSERT INTO MAP_TEST1 (A, B) VALUES (3, 'Third');
1 row inserted
```

```
SQL> CREATE STORAGE MAP MAP_TEST1_MAP FOR MAP_TEST1
cont> STORE USING (A) IN RDB$SYSTEM
cont>     WITH LIMIT OF (10);  -- can't use WITH LIMIT clause
%RDB-E-NO_META_UPDATE, metadata update failed
-RDMS-F-RELNOTEMPTY, table "MAP_TEST1" has data in it
-RDMS-E-NOCMPLXMAP, can not use complex map for non-empty table
```

8.1.48 Oracle Rdb Workload Collection Can Stop Hot Standby Replication

If you are replicating your Oracle Rdb database using the Oracle Hot Standby option, you must not use the workload collection option. By default, workload collection is disabled. However, if you enabled workload collection, you must disable it on the master database prior to performing a backup operation on that master database if it will be used to create the standby database for replication purposes. If you do not disable workload collection, it could write workload information to the standby database and prevent replication operations from occurring.

The workaround included at the end of this section describes how to disable workload collection on the master database and allow the Hot Standby software to propagate the change to the standby database automatically during replication operations.

Background Information

By default, workload collection and cardinality collection are automatically disabled when Hot Standby replication operations are occurring on the standby database. However, if replication stops (even for a brief network failure), Oracle Rdb potentially can start a read/write transaction on the standby database to write workload collection information. Then, because the standby database is no longer synchronized transactionally with the master database, replication operations cannot restart.

Note

The Oracle Rdb optimizer can update workload collection information in the RDB$WORKLOAD system table even though the standby database is opened exclusively for read-only queries. A read/write transaction is started during the disconnection from the standby database to flush the workload and cardinality statistics to the system tables.

If the standby database is modified, you receive the following messages when you try to restart Hot Standby replication operations:

```
%RDMS-F-DBMODIFIED, database has been modified; AIJ roll-forward not possible
%RMU-F-FATALRDB, Fatal error while accessing Oracle Rdb.
```

Workaround

To work around this problem, perform the following:

- On the master database, disable workload collection using the SQL clause WORKLOAD COLLECTION IS DISABLED on the ALTER DATABASE statement. For example:

```
SQL> ALTER DATABASE FILE mf_personnel
cont> WORKLOAD COLLECTION IS DISABLED;
```

This change is propagated to the standby database automatically when you restore the standby database and restart replication operations. Note that, by default, the workload collection feature is disabled. You need to disable workload collection only if you previously enabled workload collection with the WORKLOAD COLLECTION IS ENABLED clause.

- On the standby database, include the Transaction_Mode qualifier on the RMU/Restore command when you restore the standby database. You should set this qualifier to read−only to prevent modifications to the standby database when replication operations are not active. The following example shows the Transaction_Mode qualifier used in a typical RMU/Restore command:

```
$ RMU/RESTORE /TRANSACTION_MODE=READ_ONLY
              /NOCDD
              /NOLOG
              /ROOT=DISK1:[DIR]standby_personnel.rdb
              /AIJ_OPT=aij_opt.dat
              DISK1:[DIR]standby_personnel.rbf
```

If, in the future, you fail over processing to the standby database (so that the standby database becomes the master database), you can re−enable updates to the "new" master database. For example, to re−enable updates, use the SQL statement ALTER DATABASE and include the SET TRANSACTION MODES (ALL) clause. The following example shows this statement used on the new master database:

```
SQL> ALTER DATABASE FILE mf_personnel
cont> SET TRANSACTION MODES (ALL);
```

8.1.49 RMU Convert Command and System Tables

When the RMU Convert command converts a database from a previous version to Oracle Rdb V7.0 or higher, it sets the RDB$CREATED and RDB$LAST_ALTERED columns to the timestamp of the convert operation.

The RDB$xxx_CREATOR columns are set to the current user name (which is space filled) of the converter. Here xxx represents the object name, such as in RDB$TRIGGER_CREATOR.

The RMU Convert command also creates the new index on RDB$TRANSFER_RELATIONS if the database is transfer enabled.

8.1.50 Converting Single−File Databases

Because of a substantial increase in the database root file information for Release 7.0, you should ensure that you have adequate disk space before you use the RMU Convert command with single−file databases and Release 7.0 or higher.

The size of the database root file of any given database will increase a minimum of 13 blocks and a maximum of 597 blocks. The actual increase depends mostly on the maximum number of users specified for the database.

8.1.51 Restriction when Adding Storage Areas with Users Attached to Database

If you try to interactively add a new storage area where the page size is smaller than the smallest existing page size and the database has been manually opened or users are active, the add operation fails with the following error:

```
%RDB-F-SYS_REQUEST, error from system services request
-RDMS-F-FILACCERR, error opening database root DKA0:[RDB]TEST.RDB;1
-SYSTEM-W-ACCONFLICT, file access conflict
```

You can make this change only when no users are attached to the database and, if the database is set to OPEN IS MANUAL, the database is closed. Several internal Oracle Rdb data structures are based on the minimum page size, and these structures cannot be resized if users are attached to the database.

Furthermore, because this particular change is not recorded in the AIJ file, any recovery scenario will fail. Note also that if you use .aij files, you must backup the database and restart after-image journaling because this change invalidates the current AIJ recovery.

8.1.52 Support for Single-File Databases to be Dropped in a Future Release

Oracle Rdb currently supports both single-file and multifile databases on OpenVMS. However, single-file databases will not be supported in a future release of Oracle Rdb. At that time, Oracle Rdb will provide the means to easily convert single-file databases to multifile databases.

Oracle recommends that users with single-file databases perform the following actions:

- Use the Oracle RMU commands, such as Backup and Restore, to make copies, back up, or move single-file databases. Do not use operating system commands to copy, back up, or move databases.
- Create new databases as multifile databases even though single-file databases are supported in Oracle Rdb release 6.1 and release 7.0.

8.1.53 DECdtm Log Stalls

Resource managers using the DECdtm services can sometimes suddenly stop being able to commit transactions. If Oracle Rdb is installed and transactions are being run, an RMU Show command on the affected database will show transactions as being "stalled, waiting to commit".

Refer to the DECdtm documentation and release notes for information on symptoms, fixes, and workarounds for this problem. One workaround, for OpenVMS V5.5-x, is provided here.

On the affected node while the log stall is in progress, type the following command from a privileged account:

```
$ MCR LMCP SET NOTIMEZONE
```

This should force the log to restart.

This stall occurs only when a particular bit in a pointer field becomes set. To see the value of the pointer field, enter the following command from a privileged account (where <nodename> is the SCS node name of the node in question).

```
$ MCR LMCP DUMP/ACTIVE/NOFORM SYSTEM$<nodename>
```

This command displays output similar to the following:

```
Dump of transaction log SYS$COMMON:[SYSEXE]SYSTEM$<nodename>.LM$JOURNAL;1
End of file block 4002 / Allocated 4002
Log Version 1.0
Transaction log UID:   29551FC0-CBB7-11CC-8001-AA000400B7A5
Penultimate Checkpoint: 000013FD4479 0079
Last Checkpoint:        000013FDFC84 0084

Total of 2 transactions active, 0 prepared and 2 committed.
```

The stall will occur when bit 31 of the checkpoint address becomes set, as this excerpt from the previous example shows:

```
        Last Checkpoint:        000013FDFC84 0084
                                    ^
                                    |
```

When the number indicated in the example becomes 8, the log will stall. Check this number and observe how quickly it grows. When it is at 7FFF, frequently use the following command:

```
$ MCR LMCP SHOW LOG /CURRENT
```

If this command shows a stall in progress, use the workaround to restart the log.

See your HP Computer Corporation representative for information about patches to DECdtm.

8.1.54 Cannot Run Distributed Transactions on Systems with DECnet/OSI and OpenVMS Alpha Version 6.1 or OpenVMS VAX Version 6.0

If you have DECnet/OSI installed on a system with OpenVMS Alpha Version 6.1 or OpenVMS VAX Version 6.0, you cannot run Oracle Rdb operations that require the two−phase commit protocol. The two−phase commit protocol guarantees that if one operation in a distributed transaction cannot be completed, none of the operations is completed.

If you have DECnet/OSI installed on a system running OpenVMS VAX Version 6.1 or higher or OpenVMS Alpha Version 6.2 or higher, you can run Oracle Rdb operations that require the two−phase commit protocol.

For more information about the two−phase commit protocol, see the *Oracle Rdb Guide to Distributed Transactions*.

8.1.55 Multiblock Page Writes May Require Restore Operation

If a node fails while a multiblock page is being written to disk, the page in the disk becomes inconsistent and is detected immediately during failover. (Failover is the recovery of an application by restarting it on another computer.) The problem is rare and occurs because only single−block I/O operations are guaranteed by OpenVMS to be written atomically. This problem has never been reported by any customer and was detected only during stress tests in our labs.

Correct the page by an area−level restore operation. Database integrity is not compromised, but the affected area will not be available until the restore operation completes.

A future release of Oracle Rdb will provide a solution that guarantees multiblock atomic write operations. Cluster failovers will automatically cause the recovery of multiblock pages, and no manual intervention will be required.

8.1.56 Replication Option Copy Processes Do Not Process Database Pages Ahead of an Application

When a group of copy processes initiated by the Replication Option (formerly Data Distributor) begins running after an application has begun modifying the database, the copy processes will catch up to the application and will not be able to process database pages that are logically ahead of the application in the RDB$CHANGES system table. The copy processes all align waiting for the same database page and do not move on until the application has released it. The performance of each copy process degrades because it is being paced by the application.

When a copy process completes updates to its respective remote database, it updates the RDB$TRANSFERS system table and then tries to delete any RDB$CHANGES rows not needed by any transfers. During this process, the RDB$CHANGES table cannot be updated by any application process, holding up any database updates until the deletion process is complete. The application stalls while waiting for the RDB$CHANGES table. The resulting contention for RDB$CHANGES SPAM pages and data pages severely impacts performance throughput, requiring user intervention with normal processing.

This is a known restriction in Release 4.0 and higher. Oracle Rdb uses page locks as latches. These latches are held only for the duration of an action on the page and not to the end of transaction. The page locks also have blocking asynchronous system traps (ASTs) associated with them. Therefore, whenever a process requests a page lock, the process holding that page lock is sent a blocking AST (BLAST) by OpenVMS. The process that receives such a blocking AST queues the fact that the page lock should be released as soon as possible. However, the page lock cannot be released immediately.

Such work requests to release page locks are handled at verb commit time. An Oracle Rdb verb is an Oracle Rdb query that executes atomically, within a transaction. Therefore, verbs that require the scan of a large table, for example, can be quite long. An updating application does not release page locks until its verb has completed.

The reasons for holding on to the page locks until the end of the verb are fundamental to the database management system.

8.1.57 SQL Does Not Display Storage Map Definition After Cascading Delete of Storage Area

When you delete a storage area using the CASCADE keyword and that storage area is not the only area to which the storage map refers, the SHOW STORAGE MAP statement no longer shows the placement definition for that storage map.

The following example demonstrates this restriction:

```
SQL> SHOW STORAGE MAP DEGREES_MAP1
     DEGREES_MAP1
 For Table:             DEGREES1
 Compression is:        ENABLED
 Partitioning is:       NOT UPDATABLE
 Store clause:          STORE USING (EMPLOYEE_ID)
            IN DEG_AREA WITH LIMIT OF ('00250')
             OTHERWISE IN DEG_AREA2

SQL> DISCONNECT DEFAULT;
SQL> -- Drop the storage area, using the CASCADE keyword.
SQL> ALTER DATABASE FILENAME MF_PERSONNEL
cont> DROP STORAGE AREA DEG_AREA CASCADE;
SQL> --
SQL> -- Display the storage map definition.
SQL> ATTACH 'FILENAME MF_PERSONNEL';
SQL> SHOW STORAGE MAP DEGREES_MAP1
     DEGREES_MAP1
 For Table:             DEGREES1
 Compression is:        ENABLED
 Partitioning is:       NOT UPDATABLE

SQL>
```

The other storage area, DEG_AREA2, still exists, even though the SHOW STORAGE MAP statement does not display it.

A workaround is to use the RMU Extract command with the Items=Storage_Map qualifier to see the mapping.

8.1.58 ARITH_EXCEPT or Incorrect Results Using LIKE IGNORE CASE

When you use LIKE...IGNORE CASE, programs linked under Oracle Rdb Release 4.2 and Release 5.1, but run under higher versions of Oracle Rdb, may result in incorrect results or %RDB−E−ARITH_EXCEPT exceptions.

To work around the problem, avoid using IGNORE CASE with LIKE, or recompile and relink under a higher version (Release 6.0 or higher.)

8.1.59 Different Methods of Limiting Returned Rows from Queries

You can establish the query governor for rows returned from a query by using the SQL SET QUERY LIMIT statement, a logical name, or a configuration parameter. This note describes the differences between the mechanisms.

- If you define the RDMS$BIND_QG_REC_LIMIT logical name or RDB_BIND_QG_REC_LIMIT configuration parameter to a small value, the query will often fail with no rows returned. The following example demonstrates setting the limit to 10 rows and the resulting failure:

```
$ DEFINE RDMS$BIND_QG_REC_LIMIT 10
$ SQL$
SQL> ATTACH 'FILENAME MF_PERSONNEL';
SQL> SELECT EMPLOYEE_ID FROM EMPLOYEES;
%RDB-F-EXQUOTA, Oracle Rdb runtime quota exceeded
-RDMS-E-MAXRECLIM, query governor maximum limit of rows has been reached
```

  Interactive SQL must load its metadata cache for the table before it can process the SELECT statement. In this example, interactive SQL loads its metadata cache to allow it to check that the column EMPLOYEE_ID really exists for the table. The queries on the Oracle Rdb system tables RDB$RELATIONS and RDB$RELATION_FIELDS exceed the limit of rows.
  Oracle Rdb does not prepare the SELECT statement, let alone execute it. Raising the limit to a number less than 100 (the cardinality of EMPLOYEES) but more than the number of columns in EMPLOYEES (that is, the number of rows to read from the RDB$RELATION_FIELDS system table) is sufficient to read each column definition.
  To see an indication of the queries executed against the system tables, define the RDMS$DEBUG_FLAGS logical name or the RDB_DEBUG_FLAGS configuration parameter as S or B.
- If you set the row limit using the SQL SET QUERY statement and run the same query, it returns the number of rows specified by the SQL SET QUERY statement before failing:

```
SQL> ATTACH 'FILENAME MF_PERSONNEL';
SQL> SET QUERY LIMIT ROWS 10;
SQL> SELECT EMPLOYEE_ID FROM EMPLOYEES;
EMPLOYEE_ID
00164
00165
   .
   .
   .
00173
%RDB-E-EXQUOTA, Oracle Rdb runtime quota exceeded
-RDMS-E-MAXRECLIM, query governor maximum limit of rows has been reached
```

  The SET QUERY LIMIT specifies that only user queries be limited to 10 rows. Therefore, the queries used to load the metadata cache are not restricted in any way.
  Like the SET QUERY LIMIT statement, the SQL precompiler and module processor command line qualifiers (QUERY_MAX_ROWS and SQLOPTIONS=QUERY_MAX_ROWS) only limit user queries.

Keep the differences in mind when limiting returned rows using the logical name RDMS$BIND_QG_REC_LIMIT or the configuration parameter RDB_BIND_QG_REC_LIMIT. They may limit more queries than are obvious. This is important when using 4GL tools, the SQL precompiler, the SQL module processor, and other interfaces that read the Oracle Rdb system tables as part of query processing.

8.1.60 Suggestions for Optimal Usage of the SHARED DATA DEFINITION Clause for Parallel Index Creation

The CREATE INDEX process involves the following steps:

1. Process the metadata.
2. Lock the index name.
   Because new metadata (which includes the index name) is not written to disk until the end of the index process, Oracle Rdb must ensure index name uniqueness across the database during this time by taking a special lock on the provided index name.
3. Read the table for sorting by selected index columns and ordering.
4. Sort the key data.
5. Build the index (includes partitioning across storage areas).
6. Write new metadata to disk.

Step 6 is the point of conflict with other index definers because the system table and indexes are locked like any other updated table.

Multiple users can create indexes on the same table by using the RESERVING table_name FOR SHARED DATA DEFINITION clause of the SET TRANSACTION statement. For optimal usage of this capability, Oracle Rdb suggests the following guidelines:

- You should commit the transaction immediately after the CREATE INDEX statement so that locks on the table are released. This avoids lock conflicts with other index definers and improves overall concurrency.
- By assigning the location of the temporary sort work files SORTWORK0, SORTWORK1, ..., SORTWORK9 to different disks for each parallel process that issues the SHARED DATA DEFINITION statement, you can increase the efficiency of sort operations. This minimizes any possible disk I/O bottlenecks and allows overlap of the SORT read/write cycle.
- If possible, enable global buffers and specify a buffer number large enough to hold a sufficient amount of table data. However, do not define global buffers larger than the available system physical memory. Global buffers allow sharing of database pages and thus result in disk I/O savings. That is, pages are read from disk by one of the processes and then shared by the other index definers for the same table, reducing the I/O load on the table.
- If global buffers are not used, ensure that enough local buffers exist to keep much of the index cached (use the RDM$BIND_BUFFERS logical name or RDB_BIND_BUFFERS configuration parameter or the NUMBER OF BUFFERS IS clause in SQL to change the number of buffers).
- To distribute the disk I/O load, place the storage areas for the indexes on separate disk drives. Note that using the same storage area for multiple indexes will result in contention during the index creation (Step 5) for SPAM pages.
- Consider placing the .ruj file for each parallel definer on its own disk or an infrequently used disk.
- Even though snapshot I/O should be minimal, consider disabling snapshots during parallel index creation.
- Refer to the *Oracle Rdb Guide to Performance and Tuning* to determine the appropriate working set values for each process to minimize excessive paging activity. In particular, avoid using working set parameters where the difference between WSQUOTA and WSEXTENT is large. The SORT utility uses the difference between these two values to allocate scratch virtual memory. A large difference (that is, the requested virtual memory grossly exceeds the available physical memory) may lead to excessive page faulting.

- The performance benefits of using SHARED DATA DEFINITION can best be observed when creating many indexes in parallel. The benefit is in the average elapsed time, not in CPU or I/O usage. For example, when two indexes are created in parallel using the SHARED DATA DEFINITION clause, the database must be attached twice, and the two attaches each use separate system resources.
- Using the SHARED DATA DEFINITION clause on a single−file database or for indexes defined in the RDB$SYSTEM storage area is not recommended.

The following table displays the elapsed time benefit when creating multiple indexes in parallel with the SHARED DATA DEFINITION clause. The table shows the elapsed time for 10 parallel process index creations (Index1, Index2,...Index10) and one process with 10 sequential index creations (All10). In this example, global buffers are enabled and the number of buffers is 500. The longest time for a parallel index creation is Index7 with an elapsed time of 00:02:34.64, compared to creating 10 indexes sequentially with an elapsed time of 00:03:26.66. The longest single parallel create index elapsed time is shorter than the elapsed time of creating all 10 of the indexes serially.

Index Create Job	Elapsed Time
Index1	00:02:22.50
Index2	00:01:57.94
Index3	00:02:06.27
Index4	00:01:34.53
Index5	00:01:51.96
Index6	00:01:27.57
Index7	00:02:34.64
Index8	00:01:40.56
Index9	00:01:34.43
Index10	00:01:47.44
All 10	00:03:26.66

8.1.61 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.

8.1.62 Considerations when Using Holdable Cursors

If your applications use holdable cursors, be aware that after a COMMIT or ROLLBACK statement is executed, the result set selected by the cursor may not remain stable. That is, rows may be inserted, updated, and deleted by other users because no locks are held on the rows selected by the holdable cursor after a commit or rollback occurs. Moreover, depending on the access strategy, rows not yet fetched may change before Oracle Rdb actually fetches them.

As a result, you may see the following anomalies when using holdable cursors in a concurrent user environment:

- If the access strategy forces Oracle Rdb to take a data snapshot, the data read and cached may be inaccurate by the time the cursor fetches the data.
  For example, user 1 opens a cursor and commits the transaction. User 2 deletes rows read by user 1 (this is possible because the read locks are released). It is possible for user 1 to report data now deleted and committed.
- If the access strategy uses indexes that allow duplicates, updates to the duplicates chain may cause rows to be skipped, or even revisited.
  Oracle Rdb keeps track of the dbkey in the duplicate chain pointing to the data that was fetched. However, the duplicates chain could be revised by the time Oracle Rdb returns to using it.

Holdable cursors are a very powerful feature for read−only or predominantly read−only environments. However, in concurrent update environments, the instability of the cursor may not be acceptable. The stability of holdable cursors for update environments will be addressed in future versions of Oracle Rdb.

You can define the logical name RDMS$BIND_HOLD_CURSOR_SNAP or configuration parameter RDB_BIND_HOLD_CURSOR_SNAP to the value 1 to force all hold cursors to fetch the result set into a cached data area. (The cached data area appears as a "Temporary Relation" in the optimizer strategy displayed by the SET FLAGS STRATEGY statement or the RDMS$DEBUG_FLAGS S flag.) This logical name or configuration parameter helps to stabilize the cursor to some degree.

8.1.63 INCLUDE SQLDA2 Statement Is Not Supported for SQL Precompiler for PL/I in Oracle Rdb Release 5.0 or Higher

The SQL statement INCLUDE SQLDA2 is not supported for use with the PL/I precompiler in Oracle Rdb Release 5.0 or higher.

There is no workaround. This problem will be fixed in a future version of Oracle Rdb.

8.1.64 SQL Pascal Precompiler Processes ARRAY OF RECORD Declarations Incorrectly

The Pascal precompiler for SQL gives an incorrect %SQL−I−UNMATEND error when it parses a declaration of an array of records. The precompiler does not associate the END statement with the record definition, and the resulting confusion in host variable scoping causes a fatal error.

A workaround for the problem is to declare the record as a type and then define your array of that type. For example:

```
    main.spa:

        program main (input,output);

        type
        exec sql include 'bad_def.pin';    !gives error
        exec sql include 'good_def.pin';   !ok
        var
            a : char;

        begin
        end.

------------------------------------------------------------------
    bad_def.pin

    x_record = record
    y  : char;
    variable_a:  array [1..50] of record
                a_fld1 : char;
                b_fld2  : record;
                            t : record
```

```
                                        v : integer;
                                end;
                        end;
                end;
        end;
    end;
 ----------------------------------------------------------------

    good_def.pin

good_rec = record
        a_fld1 : char;
        b_fld2 : record
                t : record
                        v: integer;
                end;
        end;
end;

    x_record = record
        y  : char
        variable_a : array [1..50] of good_rec;
    end;
```

8.1.65 RMU Parallel Backup Command Not Supported for Use with SLS

The RMU Parallel Backup command is not supported for use with the Storage Library System (SLS) for OpenVMS.

8.2 Oracle CDD/Repository Restrictions

This section describes known problems and restrictions in Oracle CDD/Repository Release 7.0 and earlier.

8.2.1 Oracle CDD/Repository Compatibility with Oracle Rdb Features

Some Oracle Rdb features are not fully supported by all versions of Oracle CDD/Repository. Table 8−1 shows which versions of Oracle CDD/Repository support Oracle Rdb features and the extent of support.

In Table 8−1, repository support for Oracle Rdb features can vary as follows:

- Explicit support−−−The repository recognizes and integrates the feature, and you can use the repository to manipulate the item.
- Implicit support−−−The repository recognizes and integrates the feature, but you cannot use any repository interface to manipulate the item.
- Pass−through support−−−The repository does not recognize or integrate the feature, but allows the Oracle Rdb operation to complete without aborting or overwriting metadata. With pass−through support, a CDD−I−MBLRSYNINFO informational message may be returned.

Table 8−1 Oracle CDD/Repository Compatibility for Oracle Rdb Features

Oracle Rdb Feature	Minimum Release of Oracle Rdb	Minimum Release of Oracle CDD/Repository	Support
CASE, NULLIF, and COALESCE expressions	6.0	6.1	Implicit
CAST function	4.1	7.0	Explicit
Character data types to support character sets	4.2	6.1	Implicit
Collating sequences	3.1	6.1	Explicit
Constraints (PRIMARY KEY, UNIQUE, NOT NULL, CHECK, FOREIGN KEY)	3.1	5.2	Explicit
CURRENT_DATE, CURRENT_TIME, and CURRENT_TIMESTAMP functions	4.1	7.0	Explicit
CURRENT_USER, SESSION_USER, SYSTEM_USER functions	6.0	7.0	Explicit
Date arithmetic	4.1	6.1	Pass−through
DATE ANSI, TIME, TIMESTAMP, and INTERVAL data types	4.1	6.1	Explicit
Delimited identifiers	4.2	6.1[1]	Explicit
External functions	6.0	6.1	Pass−through
External procedures	7.0	6.1	Pass−through
EXTRACT, CHAR_LENGTH, and OCTET_LENGTH functions	4.1	6.1	Explicit
GRANT/REVOKE privileges	4.0		Pass−through

		5.0 accepts but does not store information	
Indexes	1.0	5.2	Explicit
INTEGRATE DOMAIN	6.1	6.1	Explicit
INTEGRATE TABLE	6.1	6.1	Explicit
Logical area thresholds for storage maps and indexes	4.1	5.2	Pass−through
Multinational character set	3.1	4.0	Explicit
Multiversion environment (multiple Rdb versions)	4.1	5.1	Explicit
NULL keyword	2.2	7.0	Explicit
Oracle7 compatibility functions, such as CONCAT, CONVERT, DECODE, and SYSDATE	7.0	7.0	Explicit
Outer joins, derived tables	6.0	7.0	Pass−through
Query outlines	6.0	6.1	Pass−through
Storage map definitions correctly restored	3.0	5.1	Explicit
Stored functions	7.0	6.1	Pass−through
Stored procedures	6.0	6.1	Pass−through
SUBSTRING function	4.0	7.0 supports all features 5.0 supports all but 4.2 MIA features [2]	Explicit
Temporary tables	7.0	6.1	Pass−through
Triggers	3.1	5.2	Pass−through
TRUNCATE TABLE	7.0	6.1	Pass−through
TRIM and POSITION functions	6.1	7.0	Explicit
UPPER, LOWER, TRANSLATE functions	4.2	7.0	Explicit
USER function	2.2	7.0	Explict

[1]The repository does not preserve the distinction between uppercase and lowercase identifiers. If you use delimited identifiers with Oracle Rdb, the repository ensures that the record definition does not include objects with names that are duplicates except for case.
[2]Multivendor Integration Architecture (MIA) features include the CHAR_LENGTH clause and the TRANSLATE function.

8.2.2 Multischema Databases and CDD/Repository

You cannot use multischema databases with CDD/Repository and Oracle Rdb release 7.0 and earlier. This problem will be corrected in a future release of Oracle Rdb.

8.2.3 Interaction of Oracle CDD/Repository Release 5.1 and Oracle RMU Privileges Access Control Lists

Oracle Rdb provides special Oracle RMU privileges that use the unused portion of the OpenVMS access control list (ACL) to manage access to Oracle RMU operations.

You can use the RMU Set Privilege and RMU Show Privilege commands to set and show the Oracle RMU privileges. The DCL SHOW ACL and DIRECTORY/ACL commands also show the added access control information; however, these tools cannot translate the names defined by Oracle Rdb.

Note

The RMU Convert command propagates the database internal ACL to the root file for access control entries (ACEs) that possess the SECURITY and DBADM (ADMINISTRATOR) privileges.

Oracle CDD/Repository protects its repository (dictionary) by placing the CDD$SYSTEM rights identifier on each file created within the anchor directory. CDD$SYSTEM is a special, reserved rights identifier created by Oracle CDD/Repository.

When Oracle CDD/Repository executes the DEFINE REPOSITORY command, it adds (or augments) an OpenVMS default ACL to the anchor directory. Typically, this ACL allows access to the repository files for CDD$SYSTEM and denies access to everyone else. All files created in the anchor directory inherit this default ACL, including the repository database.

Unfortunately, there is an interaction between the default ACL placed on the repository database by Oracle CDD/Repository and the Oracle RMU privileges ACL processing.

Within the ACL on the repository database, the default access control entries (ACEs) that were inherited from the anchor directory will precede the ACEs added by RMU Restore. As a result, the CDD$SYSTEM identifier will not have any Oracle RMU privileges granted to it. Without these privileges, if the user does not have the OpenVMS SYSPRV privilege enabled, Oracle RMU operations, such as Convert and Restore, will not be allowed on the repository database.

The following problems may be observed by users who do not have the SYSPRV privilege enabled:

- While executing a CDO DEFINE REPOSITORY or DEFINE DICTIONARY command:
  - If the CDD$TEMPLATEDB backup (.rbf) file was created by a previous version of Oracle Rdb, the automatic RMU Convert operation that will be carried out on the .rbf file will fail because SYSPRV privilege is required.
  - If the CDD$TEMPLATEDB backup (.rbf) file was created by the current version of Oracle Rdb, the restore of the repository database will fail because the default ACEs that already existed on the repository file that was backed up will take precedence, preventing RMU$CONVERT and RMU$RESTORE privileges from being granted to CDD$SYSTEM or the user.
  - If no CDD$TEMPLATEDB is available, the repository database will be created without a template, inheriting the default ACL from the parent directory. The ACE containing all the required Oracle RMU privileges will be added to the end of the ACL; however, the preexisting default ACEs will prevent any Oracle RMU privilege from being granted.
- You must use the RMU Convert command to upgrade the database disk format to Oracle Rdb after installing Release 7.0. This operation requires the SYSPRV privilege.
  During the conversion, RMU Convert adds the ACE containing the Oracle RMU privileges at the end of the ACL. Because the repository database already has the default Oracle CDD/Repository ACL

associated with it, the Oracle CDD/Repository ACL will take precedence, preventing the granting of the Oracle RMU privileges.
- During a CDO MOVE REPOSITORY command, the Oracle RMU privilege checking may prevent the move, as the RMU$COPY privilege has not been granted on the repository database.
- When you execute the CDD template builder CDD_BUILD_TEMPLATE, the step involving RMU Backup privilege has not been granted.

Oracle CDD/Repository Releases 5.2 and higher correct this problem. A version of the Oracle CDD/Repository software that corrects this problem and allows new repositories to be created using Oracle Rdb is provided on the Oracle Rdb kit for use on OpenVMS VAX systems. See Section 8.2.3.1 for details.

8.2.3.1 Installing the Corrected CDDSHR Images

OpenVMS VAX Systems

Note

The following procedure must be carried out if you have installed or plan to install Oracle Rdb and have already installed CDD/Repository Release 5.1 software on your system.

Due to the enhanced security checking associated with Oracle RMU commands in Oracle Rdb on OpenVMS VAX, existing CDDSHR images for CDD/Repository Release 5.1 must be upgraded to ensure that the correct Oracle RMU privileges are applied to newly created or copied repository databases.

Included in the Oracle Rdb for OpenVMS VAX distribution kit is a CDD upgraded image kit, called CDDRDB042, that must be installed after you have installed the Oracle Rdb for OpenVMS VAX kit.

This upgrade kit should be installed by using VMSINSTAL. It automatically checks which version of CDDSHR you have installed and replaces the existing CDDSHR.EXE with the corrected image file. The existing CDDSHR.EXE will be renamed SYS$LIBRARY:OLD_CDDSHR.EXE.

The upgrade installation will also place a new CDD_BUILD_TEMPLATE.COM procedure in SYS$LIBRARY for use with CDD/Repository V5.1.

Note

If you upgrade your repository to CDD/Repository V5.1 after you install Oracle Rdb V7.0, you must install the corrected CDDSHR image again to ensure that the correct CDDSHR images have been made available.

The CDD/Repository upgrade kit determines which version of CDD/Repository is installed and replaces the existing CDDSHR.EXE with the appropriate version of the corrected image.

8.2.3.2 CDD Conversion Procedure

OpenVMS VAX Systems

Oracle Rdb provides RDB$CONVERT_CDD$DATABASE.COM, a command procedure that both corrects the anchor directory ACL and performs the RMU Convert operation. The command procedure is located in SYS$LIBRARY.

Note

You must have SYSPRV enabled before you execute the procedure RDB$CONVERT_CDD$DATABASE.COM because the procedure performs an RMU Convert operation.

Use the procedure RDB$CONVERT_CDD$DATABASE.COM to process the anchor directory and update the ACLs for both the directory and, if available, the repository database.

This procedure accepts one parameter: the name of the anchor directory that contains, or will contain, the repository files. For example:

```
$ @SYS$LIBRARY:DECRDB$CONVERT_CDD$DATABASE [PROJECT.CDD_REP]
```

If many repositories exist on a system, you may want to create a DCL command procedure to locate them, set the Oracle RMU privileges ACL, and convert the databases. Use DCL commands similar to the following:

```
$ LOOP:
$       REP_SPEC = F$SEARCH("[000000...]CDD$DATABASE.RDB")
$       IF REP_SPEC .NES. ""
$       THEN
$           @SYS$LIBRARY:DECRDB$CONVERT_CDD$DATABASE    -
                'F$PARSE(REP_SPEC,,,"DIRECTORY")'
$           GOTO LOOP
$       ENDIF
```

| Contents