Oracle® Rdb7 for OpenVMS

Release Notes

Release 7.0.5

April 2000

ORACLE®

Oracle Rdb7 Release Notes, Release V7.0.5 for OpenVMS

Release 7.0.5

This document was prepared using VAX DOCUMENT Version 2.1.

Contents

B Row Cache Statements

Preface

Purpose of This Manual

This manual contains release notes for Oracle Rdb7 Release 7.0.5. The notes describe changed and enhanced features; upgrade and compatibility information; new and existing software problems and restrictions; and software and documentation corrections. These release notes cover both Oracle Rdb7 for OpenVMS Alpha and Oracle Rdb7 for OpenVMS VAX, which are referred to by their abbreviated name, Oracle Rdb7.

Intended Audience

This manual is intended for use by all Oracle Rdb7 users. Read this manual before you install, upgrade, or use Oracle Rdb7 Release 7.0.5.

Document Structure

This manual consists of twelve chapters:

Chapter 1	Describes how to install Oracle Rdb7 Release 7.0.5.
Chapter 2	Describes software errors corrected in Oracle Rdb7 Release 7.0.5.
Chapter 3	Describes software errors corrected in Oracle Rdb7 Release 7.0.4.
Chapter 4	Provides information not currently available in the Oracle Rdb7 documentation set.
Chapter 5	Describes problems, restrictions, and workarounds known to exist in Oracle Rdb7 Release 7.0.5.
Chapter 6	Describes enhancements introduced in Oracle Rdb7 Releases 7.0.5 and 7.0.4.
Chapter 7	Introduction to the new LogMiner for Oracle Rdb features available in Release 7.0.4 and beyond.
Appendix A	Describes the Row Cache feature and functionality which was added in Oracle Rdb7 Release 7.0.1.5.
Appendix B	Describes the Row Cache Statements available in Oracle Rdb7 Release 7.0.1.5 and beyond.
Appendix C	Describes software errors relating to the Row Cache feature that have been corrected in Oracle Rdb7 Release 7.0.1.5 and beyond.
Appendix D	Describes problems and restrictions relating to the Row Cache feature known to exist in Oracle Rdb7 Release 7.0.1.5 and beyond.
Appendix E	Describes the logical names relating specifically to the Row Cache feature that are available in Oracle Rdb7 Release 7.0.1.5 and beyond.

1 Installing Oracle Rdb7 Release 7.0.5

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

1.1 Requirements

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

- Oracle Rdb7 must be shutdown before you install this update kit. That is, the command file SYS$STARTUP:RMONSTOP(70).COM should be executed before proceeding with this installation. If you have an OpenVMS cluster, you must shutdown all versions of Oracle Rdb7 on all nodes in the cluster before proceeding.
- The installation requires approximately 100,000 free blocks on your system disk for OpenVMS VAX systems; 200,000 blocks for OpenVMS Alpha systems.

1.2 Invoking VMSINSTAL

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

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

variant-name

The variant names for the software update for Oracle Rdb7 Release 7.0.5 are:

- RDBSE070 for Oracle Rdb7 for OpenVMS VAX standard version.
- RDBASE070 for Oracle Rdb7 for OpenVMS Alpha standard version.
- RDBMVE070 for Oracle Rdb7 for OpenVMS VAX multiversion.
- RDBAMVE070 for Oracle Rdb7 for OpenVMS Alpha multiversion.

device-name

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

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

  ```
  DKA400:[RDBSE070.KIT]
  ```

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

  ```
  MTA0:
  ```

OPTIONS N

This parameter prints the release notes.

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

```
$ @SYS$UPDATE:VMSINSTAL RDBSE070 MTA0: OPTIONS N
```

1.3 Stopping the Installation

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

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

1.4 After Installing Oracle Rdb7

This update provides a new Oracle Rdb7 Oracle TRACE facility definition. Any Oracle TRACE selections that reference Oracle Rdb7 will need to be redefined to reflect the new facility version number for the updated Oracle Rdb7 facility definition, “RDBVMSV7.0-5”.

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

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

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

   ```
   $ LIBRARY /TEXT /EXTRACT=RDBVMSV7.0-5 -
   _$ /OUT=RDBVMS.EPC$DEF SYS$SHARE:EPC$FACILITY.TLB
   ```

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

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

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

1.5 Alpha EV67 Processor Support Added

As of this release of Rdb7, Oracle Rdb7 Release 7.0.5, the Alpha EV67 processor is supported.

1.6 Maximum OpenVMS Version Check Added

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

OpenVMS Version 7.2-n is the maximum supported version of OpenVMS.

As of Oracle Rdb7 Release 7.0.3, the Alpha EV6 processor is supported. As of Oracle Rdb7 Release 7.0.5, the Alpha EV67 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.

2
Software Errors Fixed in Oracle Rdb7 Release 7.0.5

This chapter describes software errors that are fixed by Oracle Rdb7 Release 7.0.5.

2.1 Software Errors Fixed That Apply to All Interfaces

2.1.1 Query Using Match Strategy Outline Returns Wrong Results

Bug 974665

The following query returns wrong results when the match strategy outline is used.

```
 select  s.proj_code, s.title_code, s.site_code,
         s.scan_ind, s.plan_date, t.bid_date
 from    sdp s,
         tcd t
 where   s.proj_code      = t.proj_code    and
         s.title_code     = t.title_code   and
         s.site_code      = 'CLEV'         and
         s.scan_ind       = 'Y'            and
         t.bid_date is null;
~S: Outline "QO_ZIGZAG_MATCH" used
Conjunct
Match
  Outer loop
    Sort    Conjunct        Index only retrieval of relation SDP
      Index name  SDP_IDX [1:1]
  Inner loop      (zig-zag)
    Index only retrieval of relation TCD
      Index name  TCD_IDX [1:1]
 S.PROJ_CODE   S.TITLE_CODE   S.SITE_CODE   S.SCAN_IND
   S.PLAN_DATE           T.BID_DATE
       980620259   @@@           CLEV          Y
    9-AUG-1999 00:00:00.00   NULL
1 row selected
```

The NULL predicate column "bid_date" is not part of the index SDP_IDX, which is defined as follows:

```
create index SDP_IDX on SDP
        (SITE_CODE, PLAN_DATE,
         SCAN_IND,  BID_PCT desc,
         PROJ_CODE, TITLE_CODE)
    type is SORTED;
commit work;
```

The outline is defined to use match instead of cross as follows:

```
create outline QO_ZIGZAG_MATCH
id 'CDBFC4B343409886D2FC605C40761388'
mode 0
as (
    query (
        subquery (
            SDP 0 access path index SDP_IDX
                join by match to
            TCD 1 access path index TCD_IDX
            )
        )
    )
compliance mandatory
execution options (any);
```

Workarounds for this problem are to use the SQL SET FLAGS 'NOZIGZAG_MATCH' command to turn off zigzag match or to delete the outline.

This problem has been corrected in Oracle Rdb7 Release 7.0.5.

2.1.2 Database Recovery Process Bugchecks at DIOCCHDBR$UNLATCH_GRCL + 00000398

In very rare cases of process failure when using the row cache feature, it was possible for an Oracle Rdb7 Database Recovery (DBR) to fail with an exception within DIOCCHDBR$UNLATCH_GRCL (typically at offset 00000398).

This bugcheck was due to an incorrect check while releasing a row cache latch for the failed process.

This problem has been corrected in Oracle Rdb7 Release 7.0.5. The DBR process now correctly validates the hash table slot number.

2.1.3 Dynamic Optimizer Problem with Zigzag Match

In some queries utilizing zigzag match retrieval, a problem with the interaction of the zigzag match and the dynamic optimizer may cause the query to fail to deliver appropriate records.

The queries affected contain a join of two or more tables where the optimizer has chosen to utilize a zigzag match retrieval strategy and dynamic optimization (LEAF) retrieval of data for the inner leg of the match.

The following is an example of the type of strategy associated with the affected queries.

```
    Match
      Outer loop      (zig-zag)
        Conjunct        Get     Retrieval by index of relation TABLE1
         Index name  TABLE1_INDEX_01 [1:1]
      Inner loop      (zig-zag)
        Leaf#02 Sorted TABLE2 Card=128800
          FgrNdx  TABLE2_INDEX_01 [0:0] Fan=41
          BgrNdx1 TABLE2_INDEX_02 [0:0] Fan=27
```

A problem in the delivery of data by the inner leg of the match from the dynamic optimizer data buffers prevented the appropriate match records in the outer leg from being found.

A workaround for the problem is to use the RDMS$DISABLE_ZIGZAG_MATCH logical name or the SQL SET FLAGS statement to disable zigzag match.

```
VMS> define RDMS$DISABLE_ZIGZAG_MATCH 2
```

```
or
```

```
SQL> set flags 'nozigzag_match';
```

Alternatively, dynamic optimization may be disabled by using the RDMS$MAX_ STABILITY logical name or the SQL SET FLAGS statement:

```
VMS> define RDMS$MAX_STABILITY "TRUE"
```

```
or
```

```
SQL> set flags 'max_stability';
```

This problem has been corrected in Oracle Rdb7 Release 7.0.5.

2.1.4 DBR Bugcheck in DBR$RECOVER_RCS Due to AIJ Related Database Shutdown

When the Oracle Rdb7 Row Cache Feature was enabled, the database recovery (DBR) server required that after image journaling was enabled and in a “normal” state. However, in extreme cases, such as after image journaling being shut down due to no more available journals, the DBR process would bugcheck leaving the database unuseable.

This problem has been corrected in Oracle Rdb7 Release 7.0.5. The DBR process is now more tolerant of the AIJ state.

2.1.5 Bugchecks at DIOCCH$FETCH_SNAP_SEG + 000005C4

In rare cases of process failure when using the row cache feature, it was possible for other processes to later fail with an exception within DIOCCH$FETCH_ SNAP_SEG (typically at offset 000005C4). This problem was discovered during internal testing and was not customer reported.

This bugcheck was due to incorrectly storing a snapshot page pointer in the row cache prior to writing the snapshot page back to the database. If the process failed between these two events, other users of the database could read an invalid snapshot page and this could lead to a bugcheck.

This problem has been corrected in Oracle Rdb7 Release 7.0.5. The snapshot page pointer in the row cache is not updated until the snapshot page has been written back to disk.

2.1.6 Random Corrupt Pages on Fast Processors

Bug 1142549

In rare circumstances, spurious checksum errors were being reported by Oracle Rdb. An obscure timing issue was found involving the Rdb Global Buffer feature when multiple processes attempted to access the same database pages at the same time. This error was more prevalent with the faster processors. In Oracle Rdb7 Release 7.0.3.1 and above, when this error occurs, an automatic re-read of the page obtains the correct data, and processing continues.

This problem has been corrected in Oracle Rdb7 Release 7.0.5.

2.1.7 Wrong Results from 3-Way Join Using Cross/Zigzag_Match

Bug 1041071

Wrong results were being returned from a 3-way join using the cross/zigzag match strategy.

```
select  t1.join_id,
        t2.ctrct_rvs_seq_no,
        t3.ctrct_expr_dte
    from t1, t2, t3
    where t1.id = 'V380025A' and
          t1.col2 = '01' and
          t1.col3 = 'Y' and
          t1.join_id = t2.join_id and
          t1.id = t2.id and
          t1.join_id = t3.join_id;
```

```
where the following indexes are defined :
```

```
create index t1_ndx on t1 (j_id, t1_col2, t2_col3, join_id)
create index t2_ndx on t2 (j_id, join_id, t2_col3)
create index t3_ndx on t3 (join_id)
```

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

1. t1, t2, and t3 are joined by one common segment, join_id
2. join_id is the leading segment in t3_ndx, 2nd segment in t2_ndx, and the 4th segment in t1_ndx
3. t1 and t2 are joined by j_id and join_id columns, which are leading contiguous segments in t2_ndx, but separated by 2 segments in t1_ndx
4. 2nd and 3rd segment of t1_ndx are used as equality predicates with constant values

The workaround is to define the logical RDMS$SET_FLAGS, or use the SQL SET FLAGS statement with the value NOZIGZAG_MATCH or define the logical name RDMS$DISABLE_ZIGZAG_MATCH as "2".

This problem has been corrected in Oracle Rdb7 Release 7.0.5.

2.1.8 Query Slowdown Caused by Subquery With MIN/MAX Functions

Bug 907429

A query that used to take 40 seconds to run under Rdb 6.1 and Rdb 7.0.1.1 now takes 4 hours to run under 7.0.1.2 through 7.0.2.1.

The following query is the example of the problem query, where the subqueries have aggregate functions MIN and MAX.

```
select distinct(f1.number), f1.name, f1.address, f1.dbkey from foo f1
  where (select min(f2.name) from foo f2
           where  f1.number = f2.number) <>
        (select max(f3.name) from foo f3
           where f1.number = f3.number)
order by f1.number;
```

There is no good workaround for this problem.

This problem has been corrected in Oracle Rdb7 Release 7.0.5.

2.1.9 GROUP BY/HAVING Query From a View With LIMIT TO Clause Returns Wrong Results

Bug 1204964

The following GROUP BY/HAVING query from a view with LIMIT TO clause returns wrong results.

```
SELECT seanic_month, COUNT(*) FROM A_2_view
  GROUP BY seanic_month HAVING seanic_month = ’1999 03’;
Aggregate       Conjunct        Firstn  Conjunct        Get
Retrieval sequentially of relation TABLE_2
 SEANIC_MONTH
 1999 03                 12
1 row selected
```

```
where A_2_view is defined as:
```

```
create view A_2_VIEW
    (SEANIC_MONTH) as
    select
        C1.SEANIC_MONTH
    from table_2 C1
    where ((C1.SOURCE_TABLE_TYPE = ’MY’)
        and (C1.MONTH_END_FLAG = ’Y’))
    order by C1.SEANIC_MONTH desc
    limit to 36 rows;
```

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

1. The main select query has GROUP BY/HAVING clause
2. The view is defined as a select query with LIMIT TO clause

The workaround is to remove the LIMIT TO clause from the view query.

This problem has been corrected in Oracle Rdb7 Release 7.0.5.

2.1.10 Query Returns Wrong Results When the Sequence of Same Context Predicates is Broken Up

Bug 1222168

The following query returns wrong results (should be 0 rows):

```
select h.blue
  from honda h where
        h.yellow = (select t.yellow from toyota t  limit to 1 row) and
        h.purple = 'some-color' and
        not exists (select * from animal a inner join mammal m
                    using (donkey, horse, pony)
                    where
                        m.color = h.color) and
        h.blue <> 0;
Cross block of 3 entries
  Cross block entry 1
    Aggregate       Firstn  Index only retrieval of relation toyota
      Index name  toyota_IDX1 [0:0]
  Cross block entry 2
    Leaf#01 FFirst honda Card=439
      BgrNdx1 honda_IDX1 [0:0] Bool Fan=13
  Cross block entry 3
    Conjunct        Aggregate-F1    Conjunct
    Match
      Outer loop
        Sort    Conjunct        Conjunct
        Index only retrieval of relation animal
          Index name  animal_IDX5 [0:0]
      Inner loop      (zig-zag)
        Conjunct        Index only retrieval of relation mammal
          Index name  mammal_IDX1 [0:0]
```

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

1. Two or more predicates on the same table, for example, T1
2. Followed by NOT EXISTS predicate on different tables, for example, T2 and T3
3. Followed by another predicate on T1

The workaround is to define the logical RDMS$SET_FLAGS, or use the SQL SET FLAGS statement with the value MAX_STABILITY or define the logical name RDMS$MAX_STABILITY, or group all the predicates on T1 together instead of having them separated by another predicate of a different context.

This problem has been corrected in Oracle Rdb7 Release 7.0.5.

2.1.11 Wrong Results When GROUP BY Columns are NOT Leading Subset of UNION Columns

Bug 1177495

The following GROUP BY query returns wrong results when GROUP BY columns are NOT leading subset of UNION columns:

```
SELECT  COURSE_NO, CLASS_NO, COUNT(*)
FROM
    (SELECT  Q.STDT_ID, Q.COURSE_NO, Q.CLASS_NO
     FROM
       (SELECT  STDT_ID,
                REQ_COURSE_NO AS COURSE_NO,
                REQ_CLASS_NO_1 AS CLASS_NO
             FROM    NH_TEMP A
        UNION
        SELECT  STDT_ID,
                REQ_COURSE_NO AS COURSE_NO,
                REQ_CLASS_NO_2 AS CLASS_NO
             FROM    NH_TEMP A
        ) AS Q, IN_TEMP B, PROG_TAB C
     WHERE
             B.STDT_ID = Q.STDT_ID AND
             C.PROG_ID = B.PROG_ID AND
             C.OPT_ID = B.OPT_ID) AS X
     GROUP BY COURSE_NO, CLASS_NO;
```

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

1. Select query with GROUP BY clause from a subquery with UNION legs
2. GROUP BY column order starts from the 2nd column of UNION column order

The workaround is to use UNION ALL instead of UNION.

This problem has been corrected in Oracle Rdb7 Release 7.0.5.

2.1.12 New Index Scan Algorithm Not Effective With Some Sorted Indices

Bug 1164982

In prior versions of Oracle Rdb7 (since release V7.0.1.3), the logical name RDMS$INDEX_PART_CHECK could be defined to "1" to enable a new end-of-partition checking algorithm. This new algorithm can greatly improve concurrency and performance when scanning partitioned sorted indices. The new algorithm avoids reading neighbor partitions to determine the end condition of the scan and therefore allows concurrent table processing by partitioned applications (see the PARTITION clause of the SET TRANSACTION ... RESERVING statement in the SQL Reference Manual).

However, the new algorithm was not being used when the partition USING clause listed a subset of the columns of the index. An example of such an index is shown here:

```
create unique index PERSONS_IDX
    on PERSONS (
        LAST_NAME,
        FIRST_NAME,
        MIDDLE_INITIAL
        )
    type is SORTED
    store
        using (LAST_NAME)
            in EMPIDS_LOW
                with limit of ('Dement')
            in EMPIDS_MID
                with limit of ('Myotte')
            otherwise in EMPIDS_OVER;
```

In this example, the USING clause uses only one of the columns of the index.

This problem has been corrected in Oracle Rdb7 Release 7.0.5. This release of Oracle Rdb7 now adapts correctly to this type of index definition.

Note

This new algorithm is the default behaviour in the next major release of Oracle Rdb. In that release the RDMS$INDEX_PART_CHECK algorithm is only used to disable the algorithm.

2.1.13 Wrong Results From a View Query With Left Outer Join and SUBSTRING Function

Bug 1247379

The following select query from a view with left outer join and the SUBSTRING function should return 1 row:

```
select * from V1 where VCOL2 = 'abc';
```

```
where V1 is defined as :
create view V1 (VCOL1, VCOL2, VCOL3) as
select   A1.COL1,
         substring(A1.COL2 from 1 for 3),
         A2.COL1
from
     (select COL1,COL2 from TAB1) as A1
     left outer join
     TAB2 as A2
       on A2.COL1 = A1.COL1 ;
0 rows selected
```

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

1. A view query joining 2 tables with left outer join
2. One of the view columns is a SUBSTRING function
3. The outer join table is empty
4. The selection predicate uses the column with the SUBSTRING function

The workaround is to redefine the view as a derived table, as in the following example.

```
select * from
  (select   A1.COL1,
           substring(A1.COL2 from 1 for 3),
           A2.COL1
   from
      (select COL1,COL2 from TAB1) as A1
      left outer join
      TAB2 as A2
      on A2.COL1 = A1.COL1)
      as V1 (VCOL1, VCOL2, VCOL3)
   where VCOL2 = 'abc';
```

This problem has been corrected in Oracle Rdb7 Release 7.0.5.

2.1.14 Query With EXISTS and SUBSTRING Bugchecks

Bug 1162094

The following query with EXISTS and SUBSTRING bugchecks:

```
attach 'file personnel';

create index EMP_STATUS_CODE
on EMPLOYEES(STATUS_CODE);

sel * from employees e
   where exists
     (select * from candidates c
        where e.STATUS_CODE = substring (c.CANDIDATE_STATUS from 1 for 1)
     );
```

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

1. Select query from a table with "exists" clause where the tables are joined using the equality predicate
2. The 1st table column has an index defined and the 2nd table column has no index
3. The 2nd table column has a substring function
4. The optimizer uses MATCH strategy instead of CROSS

The workaround is to define the logical RDMS$SET_FLAGS, or use the SQL SET FLAGS statement with the value NOZIGZAG_OUTER (or NOZIGZAG_MATCH).

This problem has been corrected in Oracle Rdb7 Release 7.0.5.

2.1.15 Memory Leak for Trigger Actions

Bug 1229610

In prior releases of Oracle Rdb, a small memory leak (that is allocated memory that is not released until image rundown) occurred when using domain CHECK constraints and INSERT/UPDATE statements in trigger actions.

This problem would only be noticed if the application attached to and disconnected from Rdb databases frequently or the process ran a server process that attached and disconnected often.

For the problem to occur the following must be true:

- An INSERT, DELETE, or UPDATE statement must cause a trigger to be activated
- The trigger must include an INSERT or an UPDATE on a table
- The table columns affected by these statements must be based on a domain with either an explicit domain CHECK constraint defined (possibly inherited from an Oracle CDD/Repository VALID IF definition), or an implicit precision check created for DECIMAL and NUMERIC data types when the dialect is SQL92 or ORACLE LEVEL1.

Note

This problem does not occur for column and table CHECK constraints. That is, CHECK constraints defined using the CREATE/ALTER TABLE statement.

This problem has been corrected in Oracle Rdb7 Release 7.0.5.

2.2 SQL Errors Fixed

2.2.1 Incorrect Output in SHOW STORAGE AREA (USAGE) Display

In prior versions of Oracle Rdb, the SHOW STORAGE AREA (USAGE) output did not include tables which had storage maps without STORE clauses. This meant that tables which were implicitly mapped to the default storage area were not displayed.

Additionally, global and local temporary tables, which are maintained in virtual memory and therefore not mapped to any storage area, were incorrectly listed as being mapped to the default storage area.

The following example shows these problems. In this example, the temporary table GLOBAL_TEMP_0 is listed incorrectly, and the base table BASE_TABLE_1, which should be displayed, is not.

```
SQL> create global temporary table GLOBAL_TEMP_0 (a integer)
cont>     on commit preserve rows;
SQL>
SQL> create table BASE_TABLE_1 (a integer);
SQL> create storage map BASE_TABLE_1_MAP for BASE_TABLE_1
cont>     disable compression;
SQL>
SQL> show storage area (usage) *
Storage Areas in database with filename USAGE

Database objects using Storage Area RDB$SYSTEM:
Usage            Object Name                    Map / Partition
---------------- ------------------------------ ------------------------------
Default List Area
Default Area
Table            GLOBAL_TEMP_0                  (no map)
SQL>
```

These problems have been corrected in Oracle Rdb7 Release 7.0.5. The SHOW STORAGE AREA (USAGE) display no longer includes temporary tables and now includes all implicitly mapped tables. The notation "(no store)" in the output indicates that the storage map implicitly maps the table to the default storage area.

The following examples show the revised output for the same database:

```
SQL> show storage area (usage) *
Storage Areas in database with filename USAGE

Database objects using Storage Area RDB$SYSTEM:
Usage            Object Name                    Map / Partition
---------------- ------------------------------ ------------------------------
Default List Area
Default Area
Storage Map      BASE_TABLE_1                   BASE_TABLE_1_MAP (no store)
SQL>
```

2.3 Oracle RMU Errors Fixed

2.3.1 RMU/BACKUP/AFTER/EDIT_FILE Keyword "YEAR" is Producing a Value of 1999

Bug 1135825

When using edit strings for the AIJ backup filename, and trying to get the year in the filename, the year is producing a value of "1999", but the year is "2000".

This problem *only* occurs on January 1, 2000. The year is correctly produced for December 31, 1999 as well as January 2, 2000 and all future dates.

The following example shows how to reproduce this problem, when the system date is set to January 1, 2000:

```
1.  Create an MF_PERSONNEL database
2.  Issue the following commands:

    $ rmu/set after_journal/enable/reserve=5 -
        /backup=automatic -
        /add=(name=aij1, file=task$dka0:[sqluser70.dg.aij.aij_1]aij1, -
            backup_file=task$dka0:[sqluser70.dg.aij.aij_1]aij1_bck, -
            edit_filename=(YEAR,MONTH,DAY_OF_MONTH,"-",SEQUENCE)) -
        /add=(name=aij2,file=disk$user:[dir]aij2, -
            backup_file=disk$user:[dir]aij2_bck, -
            edit_filename=(YEAR,MONTH,DAY_OF_MONTH,"-",SEQUENCE)) -
        /add=(name=aij3,file=disk$user:[dir]aij3, -
            backup_file=disk$user:[dir]aij3_bck, -
            edit_filename=(YEAR,MONTH,DAY_OF_MONTH,"-",SEQUENCE)) -
        /add=(name=aij4,file=disk$user:[dir]aij4, -
            backup_file=disk$user:[dir]aij4_bck, -
            edit_filename=(YEAR,MONTH,DAY_OF_MONTH,"-",SEQUENCE)) -
        mf_personnel

3.  Create a table called A:

    SQL> CREATE TABLE A (COLA char(3), COLB char(8));

4.  Force an AIJ journal switch-over

    $ RMU/SET AFTER/SWITCH MF_PERSONNEL

5.  Then check the AIJ backup filename for AIJ_1 once it has filled up:

    TASK> dir disk$user:[dir]*.aij

    Directory disk$user:[dir]

    AIJ1_BCK19990101-0.AIJ;1

    Total of 1 file.
```

There is no workaround to this problem.

This problem has been corrected in Oracle Rdb7 Release 7.0.5.

2.3.2 RMU/SHOW STATS "Average Per Transaction" is Relative to Epoch

The RMU Show Statistic utility displays the “Average Per Transaction” information relative to the statistics epoch, which is when the database was originally opened for statistics collection. However, it is often desireable to display information based on recent statistics collection. This is not currently possible.

The only workaround is to close and re-open the database, which is not recommended.

This problem has been corrected in Oracle Rdb7 Release 7.0.5. The RMU Show Statistic utility has been enhanced to allow the "Average Per Transaction" information to be displayed based on the last 30 statistics collections. This is known as a running average. It should be noted that the running average is computed using non-zero values (just as the normal average is computed). This means that the running average reflects the average of the most recent 30 periods of activity.

The running average display can be selected using the Tools menu, obtained using the "!" keystroke. Selecting the "Display running rate-per-sec avg" option will display the running average information. Selecting the "Display overall rate-per-sec avg" will return the display to normal. The running average display is also available during the replay of a binary input file. The screen headings will display the selected option appropriately.

2.4 Hot Standby Errors Fixed

2.4.1 Hot Standby Performance Impact on Master Database is Substantial

The Hot Standby product performance impact on the master database, and the application processing on the master database, is significantly noticeable. When measuring AIJ throughput ("AIJ blocks written"), the impact of using Oracle Rdb7 Release 7.0.3.1 Hot Standby with the "Cold" synchronization mode is approximately 35% of that when Hot Standby is not active, and the "Commit" synchronization mode is approximately 65% degradation.

This problem has been corrected in Oracle Rdb7 Release 7.0.5. Substantial performance analysis has been performed, both in controlled laboratory environments and real-world customer environments. The goal of this analysis was to identify bottlenecks and configurations that impact application processing when Hot Standby is active. As a result of this analysis, several Hot Standby algorithms, as well as general database algorithms, have been enhanced.

These enhancements result in substantial Hot Standby performance improvements. When measuring AIJ throughput ("AIJ blocks written"), the impact of using Oracle Rdb7 Release 7.0.5 Hot Standby with the "Cold" synchronization mode is approximately 5% of that when Hot Standby is not active, and the "Commit" synchronization mode is approximately 30% degradation.

Note that, with Oracle Rdb7 Release 7.0.5, the Hot Standby "Commit" synchronization mode performs *better* than the Oracle Rdb7 Release 7.0.3.1 "Cold" synchronization mode (30% versus 35%).

3 Software Errors Fixed in Oracle Rdb7 Release 7.0.4

This chapter describes software errors that are fixed by Oracle Rdb7 Release 7.0.4.

3.1 Software Errors Fixed That Apply to All Interfaces

3.1.1 RMU/LOAD into Temporary Table

Previously, RMU would not allow loading into a temporary table. While in many cases loading into a temporary table would have little value, using triggers on a temporary table may make this an attractive capability.

This restriction has been lifted in Oracle Rdb Release 7.0.4. Oracle Rdb RMU/LOAD will now load data into temporary tables. Note that the contents of the temporary table are available only to the RMU/LOAD process and will disappear when the RMU/LOAD operation completes.

3.1.2 Divide by Zero Error in Query on Large Table

Bug 800006

A simple query on a large table resulted in the following error.

```
%RDB-E-ARITH_EXCEPT, truncation of a numeric value at runtime
-SYSTEM-F-FLTDIV_F, arithmetic fault, floating divide by zero at
PC=00375C39, PSL=03C00000
```

The following is an example of the conditions needed to cause the error and the simple query used to evoke the error.

```
create table MIS_RTLSAL (RETL_CODE char(4), RETL_CREDITS integer);

create unique index MIS_RTLSAL_00 on MIS_RTLSAL (RETL_CODE);

commit;

select * from mis_rtlsal limit to 1 row;
```

For the case in which this problem was reported, the cardinality of the table was 161733114 rows and the row cluster factor for the table was 0.2081383. The large table cardinality was one of the key contributing factors. The error occurred in the Rdb Optimizer logic as it was trying to compute the cost of retrieving rows from the database.

As a workaround, this problem can be avoided by using the old cost model for the Rdb Optimizer. You can enable use of the old cost model, for example in interactive SQL, by entering the statement SET FLAGS 'OLD_COST_MODEL'; .

This problem has been corrected in Oracle Rdb7 Release 7.0.4.

3.1.3 Wrong Results With COUNT DISTINCT CASE

Bug 763963

The following query has two grouped aggregate value columns (indicated by the COUNT operation and the GROUP BY clause), a project operation (DISTINCT), and a CASE clause. These are the key factors contributing to the problem.

```
select ndate, node,
       count(distinct(case device when 'NETWORK' then process_id else null end))
         as NET_DEV_COUNT,
       count(distinct(case device when '' then process_id else null end))
         as NUL_DEV_COUNT
     from rdb_usage group by ndate, node, product
     order by ndate desc;
```

With two or more such grouped aggregate columns, the query would return wrong results for all but one of the aggregate columns. With only one grouped aggregate column in the query, the results returned were correct.

There is no known workaround for this problem.

This problem has been corrected in Oracle Rdb7 Release 7.0.4.

3.1.4 Bugcheck at RDMS$$RDMSCHEMA_UNLOAD_META+40 on Drop Area With Cascade

Bug 1022562

A problem with the way memory was allocated during the removal of a storage area caused local memory to be incorrectly overwritten which resulted in an access violation at RDMS$$RDMSCHEMA_UNLOAD_META+40.

This problem was mainly seen when at least one table spanned two or more storage areas including the storage area being dropped.

The following is an example of the storage map and alter database statement which may show this problem. In the example, a storage map is created for a table for storage across two storage areas.

```
    SQL> create storage map tab1_map for tab1
    cont>  store using ( col1 )
    cont>  in data_1 with limit of ( 1000 )
    cont>  in data_2 with limit of ( 2000 )
    cont> ;
```

If, sometime later, the database is altered to drop one of these storage areas, an access violation may occur.

```
    SQL> alter database file testdb
    cont> drop storage area data_1
    cont> cascade;
    %RDMS-I-BUGCHKDMP, generating bugcheck dump file RDSBUGCHK.DMP
    %SQL-I-BUGCHKDMP, generating bugcheck dump file SQLBUGCHK.DMP
    %SYSTEM-F-ACCVIO, access violation, reason mask=01, virtual
    address=EF9A4AC2 ...
```

A possible workaround for this problem is to alter the appropriate storage maps to exclude the storage area you wish to drop prior to altering the database. See the example below.

```
    SQL> alter storage map tab1_map
    cont>  store using ( col1 )
    cont>  in data_2 with limit of ( 2000 ) reorganize;
    SQL> alter database file testdb
    cont> drop storage area data_1
    cont> cascade;
```

This problem has been corrected in Oracle Rdb7 Release 7.0.4.

3.1.5 Unexpected I/O During DROP and TRUNCATE TABLE

Bug 989292

If a table contained one or more columns of LIST OF BYTE VARYING type, then the DROP TABLE and TRUNCATE TABLE statements would execute the equivalent of the DELETE FROM table DML statement to erase the list data from the database.

Unfortunately this meant that these statements updated the indices of the table and therefore performed unnecessary I/O to the database and journal files. In addition to this problem, TRUNCATE TABLE erroneously executed BEFORE and AFTER TRIGGER actions and some integrity constraints.

This problem has been corrected in Oracle Rdb7 Release 7.0.4. Oracle Rdb now uses a different mechanism to erase the list data which no longer causes updates to the indices, or constraint and trigger execution. The result is a significant reduction in I/O for tables containing list data. There is no change in behavior for tables that do not contain LIST OF BYTE VARYING columns.

Oracle recommends that SET TRANSACTION ... RESERVING be used to lock the table for EXCLUSIVE WRITE mode to reduce I/O, CPU and virtual memory usage during these operations. If possible, attaching to the database using the RESTRICTED ACCESS clause will further reduce I/O to the snapshot file (SNP) for the LIST STORAGE AREA. Testing of the revised algorithm for DROP TABLE showed a reduction of 10% in asynchronous reads, 82% in synchronous reads, 47% in asynchronous writes and 90% in synchronous writes when comparing the operations.

The first script uses the default reserving mode of SHARED WRITE. This will force all changes to the table to be logged to the snapshot file, and require the Rdb Server to perform row locking (or at least maintain data structures to support row locking).

```
SQL> attach 'file TEST';
SQL> set transaction read write;
SQL> drop table EMPLOYEES cascade;
SQL> commit;
```

The second script uses EXCLUSIVE WRITE to avoid the snapshot I/O for the EMPLOYEES row changes, and RESTRICTED ACCESS to eliminate snapshot I/O for the LIST storage area.

```
SQL> attach 'file TEST restricted access';
SQL> set transaction read write reserving EMPLOYEES for exclusive write
SQL> drop table EMPLOYEES cascade;
SQL> commit;
```

The reduced I/O, CPU usage and virtual memory requirements contributed to a significant reduction in elapsed time for both DROP TABLE and TRUNCATE TABLE when the table contained LIST OF BYTE VARYING columns. Improvements on specific databases will depend on database design, quantity of data, and over all system resources and therefore may vary from those reported from the Oracle Rdb test environment.

3.1.6 Incorrect Rounding of Negative Numbers in the Round Function

The Round function in SQL$FUNCTIONS.EXE or SQL$FUNCTIONSnn.EXE incorrectly rounds negative numbers. This problem has been fixed.

For example, round of (-1.56, 0) would round to -1.0 Not -2.0.

This problem has been corrected in Oracle Rdb7 Release 7.0.4.

3.1.7 Ignored Join Order Led to Poor Query Performance

The following query executed in 1 second with Oracle Rdb7 Release 7.0.1.4. In some later version (tested using Oracle Rdb7 Release 7.0.3.1), the query completed after 9 minutes. Here is the query.

```
select ZM.TEISEI_KGO, PM.PM_ST, PM.OK_ST
    from
        PM, PM_ZUMEN PZ, ZUMEN ZM
    where
        PM.HINBAN     = '009627401'   and
        PZ.HINBAN     = PM.HINBAN     and
        ZM.ZUMEN_NO   = PZ.ZUMEN_NO   and
        ZM.VER        = PZ.VER        and
        ZM.TEISEI_KGO = (select max(TEISEI_KGO) from ZUMEN ZM2
                         where  ZM2.ZUMEN_NO = ZM.ZUMEN_NO  and
                                ZM2.VER      = ZM.VER);
```

Using interactive SQL, for example, one could compare the Optimizer query strategy and cost estimates by entering the SET FLAGS 'STRATEGY,ESTIMATE' statement before executing the query. The cost estimate for the V7.0.1.4 query strategy was less than that of the V7.0.3.1 chosen strategy. The good strategy joined the tables in the following order.

PZ.PM_ZUMEN - ZM.ZUMEN - ZM2.ZUMEN - PM.PM

The poor strategy joined the tables in the following order.

ZM.ZUMEN - ZM2.ZUMEN - PZ.PM_ZUMEN - PM.PM

The Optimizer under Oracle Rdb7 Release 7.0.3.1 was ignoring the good join order. That is, the optimizer did not consider the good join order as providing a possible solution for the query strategy.

As a workaround, you can force Rdb to use the good query solution by creating a query outline under Oracle Rdb7 Release 7.0.1.4 or earlier and then applying that outline to Oracle Rdb7 Release 7.0.3.1.

This problem has been corrected in Oracle Rdb7 Release 7.0.4.

3.1.8 GROUP BY Query on a Distinct Subquery Returns Wrong Results

Bug 1089991

The following query, using match strategy, returns the wrong results.

```
select n1, n5, count(*)
    from
        (select distinct C1.N1, C1.N2, C1.N3, C2.N5
            from T1 C1, T2 C2
            where (C1.N3 = C2.N4))
            as v1 (n1, n2, n3, n5)
    group by n1,n5;
Aggregate
Merge of 1 entries
  Merge block entry 1
  Reduce  Sort    Conjunct
  Match
    Outer loop
      Sort    Get     Retrieval sequentially of relation T2
    Inner loop
      Temporary relation      Sort    Get
      Retrieval sequentially of relation T1
 N1             N5
 val2           val5                    1
 val3           val5                    1
 val1           val5                    1
 val4           val5                    1
 val1           val5                    1
 val2           val5                    1
6 rows selected
```

Where t1 and t2 are defined as follows:

```
create table T1 (
    N1  CHAR (12),
    N2  INTEGER,
    N3  CHAR (4));

create table T2 (
    N4  CHAR (4),
    N5  CHAR (12));

commit work;

insert into T1 value ('val2', 1001, '5124');
insert into T1 value ('val3', 1002, '5124');
insert into T1 value ('val1', 1003, '5159');
insert into T1 value ('val2', 1004, '5159');
insert into T1 value ('val1', 1005, '5163');
insert into T1 value ('val2', 1006, '5163');
insert into T1 value ('val1', 1007, '5152');
insert into T1 value ('val2', 1008, '5152');
insert into T1 value ('val1', 1009, '5144');
insert into T1 value ('val4', 1009, '5144');

insert into T2 value ('5124', 'val5');
insert into T2 value ('5163', 'val5');
insert into T2 value ('5144', 'val5');
```

This problem is introduced by the redundant sort elimination enhancement made in an earlier release of Oracle Rdb7. The Optimizer eliminates the GROUP BY sort as redundant as follows.

```
By combining the GROUP BY sort (C1.N1, C2.N5) and
          DISTINCT sort (C1.N1, C1.N2, C1.N3, C2.N5) into
                              (C1.N1, C2.N5, C1.N2, C1.N3)

Later the match strategy, using the join column (C1.N3 = C2.N4),
changes into (C1.N3, C2.N5, C1.N2, C1.N1) and thus produces the wrong
order for the GROUP BY operation.
```

The fix restores the GROUP BY sort to produce the correct result.

There is no workaround for this problem.

This problem has been corrected in Oracle Rdb7 Release 7.0.4.

3.1.9 After Image Journal File Format Change

With the new support for the LogMiner(tm) for Oracle Rdb feature, the After Image Journal (AIJ) internal file format minor version number has been updated for Release 7.0.4. If you enable the LogMiner for Oracle Rdb, After Image Journal files created by this version of Oracle Rdb7 may not be accepted by prior versions of Oracle Rdb7.

For this reason, you should make certain to verify and then backup your database(s) and AIJ file(s) before upgrading to Oracle Rdb7 Release 7.0.4.

3.1.10 ORDER BY Ignored in Query With a Sub-select Statement

Bug 1073357

The following query, having an explicit ORDER BY clause, would return rows in the wrong order.

```
select u.user_id, u.user_full_name,
  (select group_id from user_group_usgr gr
     where gr.user_id = u.user_id and gr.user_id = ’LEAM’)
  from user_user u
  order by u.user_id;
```

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

1. an ORDER BY clause for the outer select statement
2. a sub-select statement with its own WHERE clause
3. a portion of the WHERE clause in the form: column = ’literal-value’ (in this example: gr.user_id = ’LEAM’)
4. the referenced column (gr.user_id) is the same, though possibly from a different table, as the one named in the ORDER BY clause (u.user_id).

Given these conditions, past versions of Oracle Rdb would ignore the ORDER BY clause. Oracle Rdb assumed that the ordering was being done on a single value (in this case the value ’LEAM’). There is no known workaround for this problem.

This problem has been corrected in Oracle Rdb7 Release 7.0.4.

3.1.11 Query With Sort/Forward Scan Instead of Reverse Scan Slows Down

Bug 901904

The following query slows down drastically in Oracle Rdb7 due to the Sort /Forward strategy as compared to Oracle Rdb V6.1 where the reverse scan is applied.

```
select * from t1  where
    c4 >= 500000 and
    c1 =’10’ and
    c2 =’460’ and
    c3 =’01’
  order by c4 desc;
Firstn  Sort
Leaf#01 BgrOnly T1 Card=9022
  BgrNdx1 T1_NDX [1:0] Bool Fan=12
```

The following is the strategy output in Oracle Rdb V6.1.

```
Firstn  Conjunct        Get     Retrieval by index of relation T1
  Index name  T1_NDX [1:0] Bool         Reverse Scan
```

The table and index are defined as follows:

```
create table T1 (
    tsn integer,
    c1 char (2),
    c2 char (3),
    c3 char (2),
    c4 integer);
```

```
create index T1_NDX on T1 (c4, c1, c2, c3);
```

```
commit work;
```

Oracle Rdb7 uses the new cost model where the index scan cost increases significantly (in the range of 10 to 20 times compared to old cost model) in order to reflect the more accurate rate of I/O retrievals.

Reverse scan overhead cost depends on the forward scan index cost since it is estimated as 10% of that cost, but sort cost is estimated based on the cardinality of the tables and some startup fixed cost.

Consequently, the query will select sort and forward scan strategy over reverse scan since sort cost becomes less expensive than reverse scan and thus the sort suffers performance degradation at run time.

This fix may have a wide impact on other queries where the match with a combination of sort is chosen over the cross. This fix may now revert the query back to cross strategy due to the more expensive costing of sort.

The workaround would be to use the old cost model.

This problem has been corrected in Oracle Rdb7 Release 7.0.4.

3.1.12 Query With Selection Predicates Over UNION Legs Returns Wrong Results

Bug 1030588

The following view query with selection predicates should return 0 rows.

```
select count(*) from v1
    where
     trade_date = 36527 and
     currency      =  'EUR' ;
Aggregate       Reduce  Sort
Merge of 2 entries
  Merge block entry 1
  Conjunct
  Match
    Outer loop
      Sort Get Retrieval sequentially of relation T1 <=== missing conjunct
    Inner loop
      Get     Retrieval by index of relation T2 <=== missing conjunct
        Index name  T2_IDX [0:0]
  Merge block entry 2
  Conjunct        Get     Retrieval sequentially of relation T1
     2
1 row selected
```

Where the view v1 is defined as follows:

```
create view v1 (trade_date, currency) as
     select      p.settle_date, c.currency
         from    T1 c, T2 p
         where   (c.tradenum = p.tradenum)
  union
          select  trade_date, currency
          from    T1
          ;
```

A fix for bug 548011 was made in Oracle Rdb7 Release 7.0.1.6 to push down the selection predicates into the union legs but the fix introduced this problem.

The above query should push the predicates "trade_date = 36527" and "currency = 'EUR'" into the Merge blocks (union legs).

There is no workaround for this problem.

This problem has been corrected in Oracle Rdb7 Release 7.0.4.

3.1.13 Left Outer Join View Query With CASE Statement Returns Wrong Results

Bug 1033975

The following left outer join view query with CASE statement returns the wrong result (0 rows).

```
select buys, trade_date, update_type, portfolio, region
    from view1 where
    region =  'E'  and
    portfolio = 'JOHNE'
    and buys > 0;
Reduce  Sort    Conjunct
Cross block of 2 entries        (Left Outer Join)
  Cross block entry 1
    Conjunct
    Cross block of 2 entries        (Left Outer Join)
      Cross block entry 1
        Conjunct        Get
        Retrieval by index of relation T1
          Index name  T1_NDX [1:1] Bool
      Cross block entry 2
        Conjunct                <=== this extra conjunct is causing the problem
        Merge of 1 entries
          Merge block entry 1
          Conjunct        Index only retrieval of relation T2
            Index name  T2_NDX [1:1]
  Cross block entry 2
    Conjunct        Conjunct
    Index only retrieval of relation T3
      Index name  T3_NDX [1:1]
0 row selected
```

Where view1 is defined as follows:

```
create view view1 (
        region,
        trade_date,
        update_type,
        portfolio,
        buys ) AS
select
     c2.region,
     c2.trade_date,
     c2.update_type,
     c2.portfolio,
     c2.buys
         from view2 as c2
         left outer join
         t3 as c3 on (c2.region = c3.region)
     group by
        c2.region,
        c2.trade_date,
        c2.update_type,
        c2.portfolio,
        c2.buys ;
```

and view2 is defined as follows:

```
create view view2 (
        region,
        trade_date,
        update_type,
        portfolio,
        buys ) as
  select
     c1.region,
     c1.trade_date,
     c1.update_type,
     c1.portfolio,
     case
       when (c1.recommendation > c1.held_when_recommended)
         then c1.recommendation
         else 0
     end
     from t1 as c1
         left outer join
         (select c5.region, c5.trade_date
             from t2 c5) as c4 ( f1, f2)
          on (c1.region = c4.f1) ;
```

In a previous release of Oracle Rdb7, a fix for bug 767931 was included where the extra conjunct was generated for a left outer join query. This is usually not a problem except when a CASE statement is used in the 2nd view to further qualify the column of the selection predicates as shown above.

In the example, the conjunct of "buys" selection predicate requires the table T1 and correctly generates it in the 1st leg of the left outer join query but then it incorrectly generates it again in the 2nd cross leg where only the T2 table is available.

There is no known workaround for this problem.

This problem has been corrected in Oracle Rdb7 Release 7.0.4.

3.1.14 Query Slower Using Cross Strategy and Outline Fails to Restore to Match

Bugs 1066620 and 1066599

The following query once used a match strategy and performed well. After Oracle Rdb7 Release 7.0.1.4, the strategy changed to a cross and the query ran much slower. A query outline also failed to force the use of a match over cross strategy.

```
 select s.subject_code, s.date_audit, s.audit_username
    from  subjects_d_audit s
    where s.date_audit =
        (select max(s1.date_audit) from subjects_d_audit s1
           where s1.subject_code = s.subject_code
                 )
        and exists
            (select s2.subject_code from subjects s2
                where s2.subject_code = s.subject_code
                      )
 ;
~S: Outline "QO_A115A0E044FFD4BF_00000000" used
~S: Full compliance with the outline was not possible
%RDMS-F-OUTLINE_FAILED, could not comply with mandatory query outline directives
```

Here is the modified outline.

```
create outline QO_A115A0E044FFD4BF_00000000
id 'A115A0E044FFD4BF3957A95575671E8C'
mode 0
as (
  query (
-- For loop
    subquery (
      SUBJECTS_D_AUDIT 0      access path sequential
!        join by cross to
        join by match to
      subquery (
        SUBJECTS_D_AUDIT 1      access path sequential
        )
        join by cross to
      subquery (
        SUBJECTS 2      access path index       SUBJECTS_PKEY
        )
      )
    )
  )
compliance mandatory ;
```

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

1. the main select query with 2 or more subquery's in the where clause
2. each subquery is joined to the common column of the main context

Oracle Rdb7 Release 7.0.1.5 introduced a problem when a fix was made for Problem Report 771079.

There is no known workaround for this problem.

This problem has been corrected in Oracle Rdb7 Release 7.0.4.

3.2 SQL Errors Fixed

3.2.1 Unexpected UNSDATASS Error Reported by SQL Precompiler and Module Language

Bugs 951824 and 1033571

The DATE VMS data type included from the Oracle CDD/Repository was not correctly handled by the CAST function within the SQL precompiler and module language compilers. This resulted in the following error.

```
$ sql$pre/cobol/copy_dict /list/copy_list test.sco
                          WHERE COL2 = CAST( :F_DATE_VMS AS DATE ANSI )
                                       1
%SQL-F-UNSDATASS, (1) Unsupported date/time assignment from
F_DATE_VMS to <cast type>
```

This problem has been corrected in Oracle Rdb7 Release 7.0.4.

3.2.2 SQL IMPORT No Longer Evaluates Table and Column Constraints

In prior versions of Oracle Rdb, the SQL IMPORT statement would validate each constraint as it was applied to the recreated tables in the database. This could be a time consuming step during IMPORT requiring multiple scans of the source table.

This problem has been corrected in Oracle Rdb7 Release 7.0.4. The SQL IMPORT statement no longer requires that constraints be validated. Eliminating this step should reduce the time taken to IMPORT a database containing many constraints. Oracle recommends using RMU/VERIFY/CONSTRAINTS to check constraints.

Note

Users of the RDO IMPORT command are encouraged to use the SQL IMPORT to benefit from this change in behavior.

3.2.3 Unexpected INVACC_OUT_PARA Error Generated by CREATE MODULE

In previous versions of Oracle Rdb7, the CALL statement in a stored procedure or function might cause CREATE MODULE to fail unexpectedly.

The following example shows the error which may be generated by the CREATE MODULE statement.

```
SQL> create module SAMPLE_MODULE_P
cont>     language SQL
cont>
cont>     procedure P1 (out :a integer);
cont>     set :a = 0;
cont>
cont> end module;
SQL>
SQL> create module SAMPLE_MODULE_Q
cont>     language SQL
cont>
cont>     procedure Q1 (out :c integer);
cont>     call P1 (:c);
cont>
cont> end module;
%RDB-E-NO_META_UPDATE, metadata update failed
-RDB-E-INVALID_BLR, request BLR is incorrect at offset 58
-RDMS-E-INVACC_OUT_PARA, attempt to read from an OUT parameter
```

This error is generated when a parameter declared as OUT is passed to a stored procedure that similarly expects an OUT parameter. Oracle Rdb was incorrectly requiring IN access to the parameter.

As a workaround, the parameter may be declared as INOUT to avoid this error.

This problem has been corrected in Oracle Rdb7 Release 7.0.4. The Oracle Rdb Server now correctly checks the parameter mode and no longer requires the parameter to be declared as INOUT in this case.

3.2.4 Changed Behavior for CAST of Date/Time Values With Seconds Field

Bug 1075663

On most VAX and Alpha AXP hardware, the VMS system time is maintained to at least 1 millisecond intervals, which is more precise than is currently supported by Oracle Rdb.

Applications which accept the date/time using system services (SYS$GETTIM, SYS$BINTIM, etc) and insert those values using SQL must be aware that these date/time values will be truncated to 100th of a second by formatting routines such as SYS$ASCTIM, LIB$FORMAT_DATE_TIME and the SQL statements SELECT, PRINT, etc.

When the displayed results are subsequently used as input to queries it is possible that no matches will be found because the values do not include the full fractional seconds precision. The following example shows the potential problem.

```
SQL> select ts_col from ts_table;
 TS_COL
 10-DEC-1999 10:27:10.80
 10-DEC-1999 10:27:11.21
 10-DEC-1999 10:27:11.21
 10-DEC-1999 10:27:11.21
 10-DEC-1999 10:27:11.21
 10-DEC-1999 10:27:11.21
 10-DEC-1999 10:27:11.22
 10-DEC-1999 10:27:11.22
 10-DEC-1999 10:27:11.22
9 rows selected
SQL> select * from ts_table
cont> where ts_col = date vms'10-DEC-1999 10:27:10.80';
0 rows selected
SQL>
```

Oracle Rdb currently only supports times and timestamps up to 100ths of a second precision, e.g. TIMESTAMP(2). Starting with Rdb Version 5.1, all Rdb Server generated timestamps (CURRENT_TIME and CURRENT_TIMESTAMP) are automatically truncated to 100ths of a second. Oracle therefore recommends that these functions be used in preference to the OpenVMS system services to avoid this problem.

However, if timestamp values must be derived from an external source then care must be taken to query or store those values with correct truncation of the fractional seconds precision.

Displaying full precision of seconds

The run-time library routine LIB$FORMAT_DATE_TIME can be used to format the higher precision seconds fields. This routine is used by interactive SQL for DATE VMS types. First a date formatting logical names must be defined which includes the higher precision, as in the following example.

```
$ DEFINE/EXEC/TABLE=LNM$DT_FORMAT_TABLE -
    LIB$TIME_FORMAT_502 "!H02:!M0:!S0.!C7"
```

Once this logical is defined it can be used by any application which formats using LIB$FORMAT_DATE_TIME.

```
SQL> set date format date 1, time 502
SQL> select ts_col from ts_table;
 TS_COL
 10-DEC-1999 10:27:10.8064904
 10-DEC-1999 10:27:11.2175969
 10-DEC-1999 10:27:11.2185734
 10-DEC-1999 10:27:11.2195499
 10-DEC-1999 10:27:11.2195499
 10-DEC-1999 10:27:11.2195499
 10-DEC-1999 10:27:11.2205264
 10-DEC-1999 10:27:11.2205264
 10-DEC-1999 10:27:11.2205264
9 rows selected
SQL> select * from ts_table
cont> where ts_col between date vms'10-DEC-1999 10:27:10.80'
cont> and date vms'10-DEC-1999 10:27:10.81';
 TS_COL
 10-DEC-1999 10:27:10.8064904
1 row selected
SQL>
```

Oracle Rdb7 Release 7.0.4 has been enhanced so that the CAST operator now efficiently truncates these extra fractional seconds precision when you use the same input and output data types. That is, if you cast a DATE VMS type to DATE VMS, the fractional seconds precision is enforced. The same is true for TIME, TIMESTAMP and INTERVAL types which include the SECOND field.

The following example shows the query result after truncation using CAST function.

```
SQL> select cast(ts_col as date vms) from ts_table;

 10-DEC-1999 10:27:10.8000000
 10-DEC-1999 10:27:11.2100000
 10-DEC-1999 10:27:11.2100000
 10-DEC-1999 10:27:11.2100000
 10-DEC-1999 10:27:11.2100000
 10-DEC-1999 10:27:11.2100000
 10-DEC-1999 10:27:11.2200000
 10-DEC-1999 10:27:11.2200000
 10-DEC-1999 10:27:11.2200000
9 rows selected
SQL> select * from ts_table
cont> where cast(ts_col as date vms) = date vms'10-DEC-1999 10:27:10.80';
 TS_COL
 10-DEC-1999 10:27:10.8064904
1 row selected
SQL>
```

3.2.5 SQL Rejects Queries Which Use Column Named VALUE

Bug 1149113

In prior versions of Oracle Rdb, using a column named VALUE was prohibited because of the special nature of this keyword. VALUE is a special identifier reserved for use in a domain CHECK constraint definition. Attempts to use such a column caused a fatal error for DML statements (INSERT, SELECT, DELETE and UPDATE) as shown in this simple example.

```
SQL> select value from v;
%SQL-F-VALUEILL, VALUE cannot be used outside of a domain constraint
```

While it is true that VALUE is a reserved word in the ANSI and ISO SQL Standards, other similar keywords cause an information message to be generated so that older applications can continue to execute unchanged. However, this VALUEILL error prevented applications from working with more recent versions of Oracle Rdb.

With this release of Oracle Rdb, the VALUEILL error is no longer reported and VALUE is treated in the same way as other reserved words. That is, a warning is issued by default. The query will fail if a dialect is established such as SQL92.

```
SQL> select value from v;
%SQL-I-DEPR_FEATURE, Deprecated Feature: Keyword VALUE used as an identifier
0 rows selected
SQL> set dialect 'sql92';
SQL> select value from v;
%SQL-F-RES_WORD_AS_IDE, Keyword VALUE used as an identifier
```

This problem has been corrected in Oracle Rdb7 Release 7.0.4.

3.3 Oracle RMU Errors Fixed

3.3.1 RMU Extract Has Enhanced Extract of Conditional Expressions

Oracle Rdb7 Release 7.0.4 improves the extraction of the conditional expressions COALESCE, NVL, NULLIF, and simple CASE expressions.

In prior releases, these expressions were incorrectly extracted, and may have appeared as searched CASE expressions. This occurred because the pattern matching algorithm often didn't find a match for these expressions. This release enhances the pattern matching to match correctly these expressions.

The side effect of these changes is that some searched CASE expressions may be extracted as an alternate and more compact form of the conditional expression.

The following list shows the equivalent expressions matched by RMU Extract.

- 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.

3.3.2 RMU/REPLICATE AFTER START Command Fails on TCP/IP With Large Port Numbers

When the TCP/IP service for Hot Standby is defined with a port number larger than 32,767, the network connection would fail due to incorrect network port to host port translation of the port number.

```
$ rmu/replicate after_journal start m_testdb.rdb -
     /standby=node::device-directory:s_testdb.rdb
%COSI-F-CONNECFAIL, connect over network timed-out or failed
```

This problem has been corrected in Oracle Rdb7 Release 7.0.4. With this release, TCP/IP port numbers up to 65,535 will be supported.

3.3.3 SHOW STATS Cannot Replay /OPTIONS=ROW_CACHE Input File

The RMU Show Statistic utility was unable to replay binary output files created with the /OPTIONS=ROW_CACHE or /OPTIONS=ALL qualifiers. The problem only occurs when the database has row caching enabled.

The only work-around is to *not* use the /OPTIONS=ROW_CACHE qualifier.

This problem has been corrected in Oracle Rdb7 Release 7.0.4. The /OPTIONS=ROW_CACHE and /OPTIONS=ALL qualifiers now work correctly.

3.3.4 RMU/SHOW LOCKS Difficult to Identify Lock Conflict Culprit

Each line of the RMU/SHOW LOCKS utility output shows the process that is waiting and the process that is blocking it. At least one of the blocking processes is not in the list of waiting processes. In other words, the process is either running a long transaction or, more likely, it's waiting for a non-database event. If this process is terminated or forced to finish its transaction, the waiting processes start to move, and frequently the blocks all clear.

```
Waiting         Blocker
0001            0002
0002            0003
0003            0004    <-- stop/id=0004 may well free things up
0005            0003
0006            0005    <-- stop/id=0005 would only help 0006
```

Maybe processes 0001-0006 are all culprits, but there is a sense in which process 0004 is more culpable.

There is no workaround to this problem other than manually searching the RMU/SHOW LOCKS /MODE=WAITING output manually.

This problem has been corrected in Oracle Rdb7 Release 7.0.4. The RMU/SHOW LOCKS utility has been enhanced to support a new display mode, /MODE=CULPRIT.

The /MODE=CULPRIT output is a sanitized version of the /MODE=WAITING output. The /MODE=CULPRIT qualifier displays only the set of locks for processes that are blocking other processes but are themselves not blocked. This output represents the processes that are the source of database stalls and performance degradation.

In the following real-world example, one process is blocking the entire application. Compare the difference in the output between the /MODE=WAITING and /MODE=CULPRIT output.

The /MODE=WAITING qualifier displays the following output:

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

--------------------------------------------------------------------------------
Resource: record 109:1085:0

          ProcessID Process Name       Lock ID   System ID Requested Granted
          --------- ---------------    --------- --------- --------- -------
Blocker:  3C806080  RICK10.........    23002C6A  000100E4  PR        NL
Waiting:  3C806081  RICK8..........    33004A4A  000100E4  PR        NL
Waiting:  3C805C86  RICK14.........    470088DA  000100E4  PR        NL
Waiting:  3C805E7E  RICK4..........    410002FA  000100E4  PR        NL

--------------------------------------------------------------------------------
Resource: record 109:1085:0

          ProcessID Process Name       Lock ID   System ID Requested Granted
          --------- ---------------    --------- --------- --------- -------
Blocker:  3C806081  RICK8..........    33004A4A  000100E4  PR        NL
Waiting:  3C805C86  RICK14.........    470088DA  000100E4  PR        NL
Waiting:  3C805E7E  RICK4..........    410002FA  000100E4  PR        NL

--------------------------------------------------------------------------------
Resource: record 109:1085:0
```

```
          ProcessID Process Name        Lock ID   System ID Requested Granted
          --------- ---------------     --------- --------- --------- -------
Blocker:  3C806083  RICK12.........     0E008171  000100E4  PR        PR
Waiting:  3C805E82  RICK9..........     2A0087FF  000100E4  EX        PR
Waiting:  3C805A84  RICK11.........     3B00F074  000100E4  PR        NL
Waiting:  3C806085  RICK13.........     3200EF7E  000100E4  PR        NL
Waiting:  3C806080  RICK10.........     23002C6A  000100E4  PR        NL
Waiting:  3C806081  RICK8..........     33004A4A  000100E4  PR        NL
Waiting:  3C805C86  RICK14.........     470088DA  000100E4  PR        NL
Waiting:  3C805E7E  RICK4..........     410002FA  000100E4  PR        NL

--------------------------------------------------------------------------------
Resource: record 109:1085:0

          ProcessID Process Name        Lock ID   System ID Requested Granted
          --------- ---------------     --------- --------- --------- -------
Blocker:  3C805E82  RICK9..........     2A0087FF  000100E4  EX        PR
Waiting:  3C805A84  RICK11.........     3B00F074  000100E4  PR        NL
Waiting:  3C806085  RICK13.........     3200EF7E  000100E4  PR        NL
Waiting:  3C806080  RICK10.........     23002C6A  000100E4  PR        NL
Waiting:  3C806081  RICK8..........     33004A4A  000100E4  PR        NL
Waiting:  3C805C86  RICK14.........     470088DA  000100E4  PR        NL
Waiting:  3C805E7E  RICK4..........     410002FA  000100E4  PR        NL

--------------------------------------------------------------------------------
Resource: record 109:1085:0

          ProcessID Process Name        Lock ID   System ID Requested Granted
          --------- ---------------     --------- --------- --------- -------
Blocker:  3C8004DC  RICK5..........     370088C2  000100E4  PR        PR
Waiting:  3C805E82  RICK9..........     2A0087FF  000100E4  EX        PR
Waiting:  3C805A84  RICK11.........     3B00F074  000100E4  PR        NL
Waiting:  3C806085  RICK13.........     3200EF7E  000100E4  PR        NL
Waiting:  3C806080  RICK10.........     23002C6A  000100E4  PR        NL
Waiting:  3C806081  RICK8..........     33004A4A  000100E4  PR        NL
Waiting:  3C805C86  RICK14.........     470088DA  000100E4  PR        NL
Waiting:  3C805E7E  RICK4..........     410002FA  000100E4  PR        NL

--------------------------------------------------------------------------------
Resource: record 109:1085:0

          ProcessID Process Name        Lock ID   System ID Requested Granted
          --------- ---------------     --------- --------- --------- -------
Blocker:  3C805C86  RICK14.........     470088DA  000100E4  PR        NL
Waiting:  3C805E7E  RICK4..........     410002FA  000100E4  PR        NL

--------------------------------------------------------------------------------
Resource: record 109:1660:1

          ProcessID Process Name        Lock ID   System ID Requested Granted
          --------- ---------------     --------- --------- --------- -------
Blocker:  3C8004DC  RICK5..........     2D00D032  000100E4  EX        EX
Waiting:  3C806083  RICK12.........     5700D6F9  000100E4  EX        NL

--------------------------------------------------------------------------------
Resource: record 109:1085:0

          ProcessID Process Name        Lock ID   System ID Requested Granted
          --------- ---------------     --------- --------- --------- -------
Blocker:  3C806085  RICK13.........     3200EF7E  000100E4  PR        NL
Waiting:  3C806080  RICK10.........     23002C6A  000100E4  PR        NL
Waiting:  3C806081  RICK8..........     33004A4A  000100E4  PR        NL
Waiting:  3C805C86  RICK14.........     470088DA  000100E4  PR        NL
Waiting:  3C805E7E  RICK4..........     410002FA  000100E4  PR        NL

--------------------------------------------------------------------------------
Resource: record 109:1085:0
```

```
         ProcessID Process Name        Lock ID   System ID Requested Granted
         --------- ---------------     --------- --------- --------- -------
Blocker:  3C805A84  RICK11.........     3B00F074  000100E4  PR        NL
Waiting:  3C806085  RICK13.........     3200EF7E  000100E4  PR        NL
Waiting:  3C806080  RICK10.........     23002C6A  000100E4  PR        NL
Waiting:  3C806081  RICK8..........     33004A4A  000100E4  PR        NL
Waiting:  3C805C86  RICK14.........     470088DA  000100E4  PR        NL
Waiting:  3C805E7E  RICK4..........     410002FA  000100E4  PR        NL
```

The /MODE=CULPRIT qualifier displays the following output:

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

--------------------------------------------------------------------------------
Resource: record 109:1085:0
         ProcessID Process Name        Lock ID   System ID Requested Granted
         --------- ---------------     --------- --------- --------- -------
Blocker:  3C8004DC  RICK5..........     370088C2  000100E4  PR        PR
Waiting:  3C805E82  RICK9..........     2A0087FF  000100E4  EX        PR
Waiting:  3C805A84  RICK11.........     3B00F074  000100E4  PR        NL
Waiting:  3C806085  RICK13.........     3200EF7E  000100E4  PR        NL
Waiting:  3C806080  RICK10.........     23002C6A  000100E4  PR        NL
Waiting:  3C806081  RICK8..........     33004A4A  000100E4  PR        NL
Waiting:  3C805C86  RICK14.........     470088DA  000100E4  PR        NL
Waiting:  3C805E7E  RICK4..........     410002FA  000100E4  PR        NL

--------------------------------------------------------------------------------
Resource: record 109:1660:1

         ProcessID Process Name        Lock ID   System ID Requested Granted
         --------- ---------------     --------- --------- --------- -------
Blocker:  3C8004DC  RICK5..........     2D00D032  000100E4  EX        EX
Waiting:  3C806083  RICK12.........     5700D6F9  000100E4  EX        NL
```

In this example, process 3C8004DC is the culprit of two separate, but probably related, stalls.

3.3.5 RMU BACKUP to Tape Hung if Bad Checksum

Bug 1059787

When a database page contained an invalid checksum, RMU/BACKUP/ONLINE to a tape device hung instead of reporting the error if checksum checking was enabled.

The following example shows a sample RMU BACKUP command line which caused the hang if there was a bad checksum on a database page.

```
RMU/BACKUP/ONLINE/LABEL=BACK01 database.rdb TAPE:database.rbf
```

The following shows the corrected behavior: an error message is ouput and the backup to tape reports the fatal error and does not hang.

```
RMU/BACKUP/ONLINE/LABEL=BACK01 database.rdb TAPE:database.rbf
%RMU-F-CANTREADDBS, error reading pages 2:3-3
-RMU-F-CHECKSUM, checksum error - computed 67C3D4E8, page contained 00003039
%RMU-F-FATALERR, fatal error on BACKUP
```

As a workaround, to avoid the problem do not enable checksum checking for RMU BACKUP to tape.

```
RMU/BACKUP/NOCHECKSUM/ONLINE/LABEL=BACK01 database.rdb TAPE:database.rbf
```

This problem has been corrected in Oracle Rdb7 Release 7.0.4.

3.3.6 RMU BACKUP to Tape Hung on QUIT Response to Wrong Label Message

RMU BACKUP to tape devices hung when the user chose the "QUIT" response as the reply to the message output by RMU BACKUP when a label was specified in the RMU BACKUP command which did not match the label on the tape device being used for the backup.

The following example shows an RMU BACKUP command line and QUIT response to the wrong label message output by RMU BACKUP which caused RMU BACKUP to tape to hang.

```
RMU/BACKUP/REWIND/LABEL=(badlab01,badlab02)/LOADER MF_PERSONNEL.RDB -
$111$MUA30:MF_PERSONNEL.RBF/MASTER, $111$MUA31:/MASTER
%RMU-I-WRNGLBL, Tape on _$111$MUA30 was incorrectly labeled. Expected GOODLAB
 - Found BADLAB01
%RMU-I-TAPEDISPW, Specify tape disposition for _$111$MUA30 (QUIT,INITIALIZE,
 RETRY,UNLOAD)
 quit
```

The workaround for this problem is to choose an option other than "QUIT" in response to the bad label message or to reenter the RMU BACKUP command specifying a label that matches the label on the tape device.

This problem has been corrected in Oracle Rdb7 Release 7.0.4.

3.3.7 RMU/REPAIR/INIT=FREE_PAGES/ABM Did Not Return an Error

Bug 968268

The RMU/REPAIR documented restriction that the qualifiers /INITIALIZE=FREE_PAGES and /ABM were conflicting qualifiers and could not be used together on the same RMU/REPAIR command line was not enforced by a conflicting qualifiers error message but was allowed.

The following example shows that the /INITIALIZE=FREE_PAGES and /ABM qualifiers were accepted by the RMU/REPAIR command when a conflicting qualifiers error should have been returned.

```
$RMU/REPAIR/ABM/SPAM/INITIALIZE=FREE_PAGES/AREA=AREA_NAME MF_PERSONNEL
```

The following example shows that an error is now returned and the command is not accepted.

```
$RMU/REPAIR/ABM/SPAM/INITIALIZE=FREE_PAGES/AREA=AREA_NAME MF_PERSONNEL
%RMU-F-CONFLSWIT, conflicting qualifiers /ABM and /INITIALIZE=FREE_PAGES
```

As a workaround, do not include the /ABM and /INITIALIZE=FREE_PAGES qualifiers in the same RMU/REPAIR command line.

This problem has been corrected in Oracle Rdb7 Release 7.0.4.

3.3.8 Incorrect BADIDXREL Messages From Online RMU Verify

Bugs 883349 and 1039089

An online RMU VERIFY of a database index where /TRANSACTION_TYPE=READ_ONLY was specified sometimes output incorrect RMU-W-BADIDXREL warning messages when the index was being concurrently modified by other users. These same BADIDXREL messages were not output if the index was not being modified during the online verify or if READ_ONLY was not specified with the /TRANSACTION_TYPE qualifier.

The following example shows the RMU VERIFY command for verifying a database index using the /TRANSACTION_TYPE=READ_ONLY qualifier and the resulting RMU-W-BADIDXREL warning message which was not output if /TRANSACTION_TYPE=READ_ONLY was not specified.

```
$ rmu/verify/noroot/transaction_type=read_only -
/index=(db_index)/data rdb_database
%RMU-W-BADIDXREL, Index DB_INDEX either points to a non-existent record or
                  has multiple pointers to a record in table RDB_TABLE.
                  The logical dbkey in the index is 527:2324:1.
```

The workaround for this problem is to use the /TRANSACTION_TYPE=READ_ONLY qualifier when no user transaction is modifying the database index being verified or to specify another /TRANSACTION_TYPE such as PROTECTED (the default) or EXCLUSIVE.

```
$ rmu/verify/noroot/transaction_type=exclusive -
/index=(db_index)/data rdb_database
```

This problem has been corrected in Oracle Rdb7 Release 7.0.4.

3.3.9 RMU VERIFY Did Not Find a .RDA File After an RMU MOVE

RMU/VERIFY did not find a .RDA database area file which had been updated to a new version by the RMU/MOVE of the associated database snapshot file which had been executed on another node of the cluster.

The following example shows the error.

```
On Node1:
$ RMU/OPEN/ACC=UNR MF_PERSONNEL
On Node2:
$ RMU/OPEN/ACC=UNR MF_PERSONNEL
$ CREATE [.TEST] /DIR
$ RMU/MOVE/ONL/LOG MF_PERSONNEL RESUMES /SNAP=FILE=[.TEST]
On Node1:
$ RMU/VERIFY/LOG/TRANS=READ_ONLY/AREA=RESUMES/SNAP MF_PERSONNEL
%RMU-I-BGNROOVER, beginning root verification
%RMU-F-OPNFILERR, error opening file
DISK:[DIRECTORY]RESUMES.RDA;1
%RMU-F-FILNOTFND, file not found
%RMU-E-BDAREAOPN, unable to open file
DISK:[DIRECTORY]RESUMES.RDA;1 for storage area RESUMES
%RMU-F-ABORTVER,  fatal error encountered; aborting verification
```

A workaround for this problem is to do the RMU/MOVE on the same node as the RMU/VERIFY.

This problem has been corrected in Oracle Rdb7 Release 7.0.4.

3.4 Row Cache Errors Fixed

3.4.1 Row Cache Server Operator Notification

Similar to other Oracle Rdb7 database servers, the Row Cache Server (RCS) process now sends start and terminate messages to the system operator if database operator notifications are enabled.

The following example shows the format of the Row Cache Server operator message:

```
$ REPLY/ENABLE=CENTRAL
%%%%%%%%%%%  OPCOM  28-SEP-1999 17:16:57.32  %%%%%%%%%%%
Operator TTA0: has been enabled, username RC

%%%%%%%%%%%  OPCOM  28-SEP-1999 17:16:57.33  %%%%%%%%%%%
Operator status for operator TTA0:
CENTRAL

$ RMU/OPEN DUA0:[DB]MFP
%%%%%%%%%%%  OPCOM  28-SEP-1999 17:15:47.66  %%%%%%%%%%%
Message from user RDBVMS on RYEROX
Oracle Rdb X7.1-00 Database DUA0:[DB]MFP.RDB;1 Event Notification
Row Cache Server started
```

3.4.2 Row Cache Did Not Avoid Certain Database Writes

In certain situations, the Oracle Rdb7 Row Cache feature did not avoid database update I/O that it otherwise could have avoided. In particular, when a database record was modified when it was not originally in the cache, it is possible that the database page containing the row could be written back to the database where it otherwise would not have to be.

Some applications may find a performance improvement when using the Row Cache feature with Oracle Rdb7 Release 7.0.4 due to the reduction in unneeded database write I/O for some update operations.

3.4.3 RMU /CLOSE /WAIT Would Not Always Wait When Row Cache Enabled

When using the Row Cache feature with many or large caches, it was possible that the RMU /CLOSE /WAIT command could return to the user with the database still actually being shut down.

This problem has been corrected in Oracle Rdb7 Release 7.0.4. The RMU /CLOSE /WAIT command now does an additional check to make sure that the database is not open before returning to the user.

3.5 Hot Standby Errors Fixed

3.5.1 RMU/REPLICATE AFTER START Command Fails Due to Lost AIJ Write

There is a situation where Hot Standby fails but has already committed a transaction that did not get written to the AIJ journal. During re-start of Hot Standby, the lost write is before the last committed transaction causing re-start to fail. The following error message is returned during restart.

```
$ rmu/replicate after_journal start  s_testdb.rdb -
    /master_root=m_testdb.rdb
%RDMS-F-CANTSTARTLRS, error starting AIJ Log Roll-Forward Server process
%RMU-F-FATALRDB, Fatal error while accessing Oracle Rdb.
```

The following messages would appear in the LRS log file:

```
23-OCT-1999 07:49:41 - Transaction recovery not started during restart
23-OCT-1999 07:49:41 - This usually occurs when a manual roll-forward operation
23-OCT-1999 07:49:41 - using master database AIJ journals did not fully complete
23-OCT-1999 07:49:41 - This is sometimes caused by an AIJ switch-over operation
23-OCT-1999 07:49:41 - while Hot Standby is inactive

23-OCT-1999 07:49:41 - Failure reason: %RDMS-F-CANTSTARTLRS,
                            error starting AIJ Log Roll-Forward Server process
```

This problem has been corrected in Oracle Rdb7 Release 7.0.4.

4 Documentation Corrections

This chapter provides information not currently available in the Oracle Rdb7 documentation set.

4.1 Documentation Corrections

4.1.1 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;
```

4.1.2 Compressed Sorted Index Entry Stored in Incorrect Storage Area

This note was originally included in the Oracle Rdb7 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 Rdb7 Relese 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 emtpy 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 Rdb7 Release 7.0.1.3.

4.1.3 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*.

4.1.4 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.

4.1.5 Waiting for Client Lock Message

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

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

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

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

The leftmost value seen in the hexadecimal output contains the ID of the object. The following ID describes the tables, routines, modules and storage map areas.

- For tables and views, the ID represents the unique value found in the RDB$RELATION_ID column of the RDB$RELATIONS system table for the given table.
- For routines, the ID represents the unique value found in the RDB$ROUTINE_ID column of the RDB$ROUTINES system table for the given routine.
- For modules, the ID represents the unique value found in the RDB$MODULE_ID column of the RDB$MODULES system table for the given module.
- For storage map areas, the ID presents the physical area ID. The “waiting for client lock” message on storage map areas is very rare. This may be raised for databases that have been converted from versions prior to Oracle Rdb 5.1.

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

Table 4–1 Object Type Values

Object	Hexadecimal Value
Tables or views	00000004
Routines	00000006
Modules	00000015
Storage map areas	0000000E

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

The following example shows a “waiting for client” lock message from the Stall Messages screen:

```
Process.ID Since......   Stall.reason............................. Lock.ID.
46001105:2 10:40:46.38 - waiting for client '........' 000000190000000400000055
                                                    1         2        3       4
```

The following list describes each part of the client lock:

1 .... indicates nonprintable characters.

2 00000019 indicates unique identifier hex value 19 (RDB$RELATION_ID = 25).

3 00000004 indicates object type 4 which is a table.

4 00000055 indicates this is a client lock.

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

```
SQL> SELECT RDB$RELATION_NAME FROM RDB$RELATIONS WHERE RDB$RELATION_ID = 25;
SQL> SELECT RDB$MODULE_NAME FROM RDB$MODULES WHERE RDB$MODULE_ID = 12;
SQL> SELECT RDB$ROUTINE_NAME FROM RDB$ROUTINES WHERE RDB$ROUTINE_ID = 7;
```

Note

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

For more detailed lock information, perform the following steps:

1. Press the L option from the horizontal menu to display a menu of Lock IDs.
2. Select the desired Lock ID.

4.1.6 Documentation Error in *Oracle Rdb7 Guide to Database Performance and Tuning*

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

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

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

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

> **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*.

4.1.7 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 Rdb7.

This has been corrected in this release of Oracle Rdb7. This keyword is now recognized by the SET FLAGS statement. As a workaround the logical name RDMS$BIND_OUTLINE_FLAGS "I" can be used to set this attribute.

4.1.8 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 Rdb7 and is used to enable debug flags output for internal queries such as constraints and triggers. It can be used in conjunction with other options such as STRATEGY, BLR, and EXECUTION. For example, the following flag settings are equivalent to defining the RDMS$DEBUG_FLAGS as ISn and shows the strategy used by the trigge'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
```

4.1.9 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 or the UNIX environment variable SQL_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.

4.1.10 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
```

4.1.11 Undocumented SET Commands and Language Options

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

4.1.11.1 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;
```

4.1.11.2 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;
```

4.1.12 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
```

4.1.13 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.

4.1.14 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.

4.1.15 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.

4.1.16 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].

5 Known Problems and Restrictions

This chapter describes problems, restrictions, and workarounds known to exist in Oracle Rdb7 Release 7.0.5.

5.0.1 Oracle Rdb and OpenVMS ODS-5 Volumes

The OpenVMS Version 7.2 release introduced Extended File Specifications, 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 directories

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.

A future release of Oracle Rdb is expected to relax some of these restrictions and support ODS-5 volumes.

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

Bug 551240

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

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

- SQL/Services database class services. When SQL/Services (possibly accessed by ODBC) accesses a database, it allows the user to logon to the database and the SQL/Services server then impersonates that user in the database.

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

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

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

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

5.0.3 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.

5.0.4 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.

5.0.5 Unexpected DATEEQLILL Error During IMPORT With CREATE INDEX or CREATE STORAGE MAP

Bug 1094071

When the SQL IMPORT statement includes CREATE STORAGE MAP or CREATE INDEX statements which use TIMESTAMP or DATE ANSI literals in the WITH LIMIT OF clause, it fails with the following error:

```
%SQL-F-UNSDATXPR, Unsupported date expression
-SQL-F-DATEEQLILL, Operands of date/time comparison are incorrect
```

The same CREATE STORAGE MAP or CREATE INDEX statements work correctly when used outside of the IMPORT statement.

This error is generated because the SQL IMPORT statement tries to validate the data type of the column against that of the literal value. However, during this phase of the IMPORT, the table does not yet exist.

A workaround for this problem is to use DATE VMS literals in the WITH LIMIT OF clause and allow the Rdb Server to perform the data type conversion at runtime.

This restriction will be relaxed in a future version of Oracle Rdb.

5.0.6 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.

5.0.7 IMPORT Unable to Import Some View Definitions

Bug 520651

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

The following example shows the errors from IMPORT.

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

The following script can be used to demonstrate the problem.

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

create module TFORMAT
    language SQL

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

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

commit;

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

This restriction will be lifted in a future release of Oracle Rdb. Currently the workaround is to save the view definitions and reapply them after the IMPORT completes.

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

5.0.8 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.

5.0.9 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.

5.0.10 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 Rdb7 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.

5.0.11 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.

5.0.12 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.

5.0.13 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.

5.0.14 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.

5.0.15 Oracle Rdb and the SRM_CHECK Tool

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

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

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

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

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

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

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

5.0.16 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 qualifie on all of your RMU database backup commands.

5.0.17 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.

5.0.18 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 clusterwide:

```
%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.

5.0.19 SELECT Query May Bugcheck with PSII2SCANGETNEXTBBCDUPLICATE Error

Bug 683916

A bugcheck could occur when a ranked B-tree index is used in a query after a database has been upgraded to Release 7.0.1.3. This is a result of index corruption that was introduced in previous versions of Oracle Rdb7. This corruption has been fixed and indexes created using Release 7.0.1.3 will not be impacted.

As a workaround, delete the affected index and re-create it under Oracle Rdb7 Release 7.0.1.3 or later.

5.0.20 DBAPack for Windows 3.1 is Deprecated

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

5.0.21 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.

5.0.22 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.

5.0.23 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.

5.0.24 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.

5.0.25 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.

5.0.26 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.

5.0.27 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
```

5.0.28 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.

5.0.29 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.

5.0.30 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.

5.0.31 Restriction Using Backup Files Created Later than Oracle Rdb7 Release 7.0.1

Bug 521583

Backup files created using Oracle Rdb7 releases later than 7.0.1 cannot be restored using Oracle Rdb7 Release 7.0.1. To fix a problem in a previous release, some internal backup file data structures were changed. These changes are not backward compatible with Oracle Rdb7 Release 7.0.1.

If you restore the database using such a backup file, then any attempt to access the restored database may result in unpredictable behavior, even though a verify operation may indicate no problems.

There is no workaround to this problem. For this reason, Oracle Corporation strongly recommends performing a full and complete backup both before and after the upgrade from Release 7.0.1 to later releases of Oracle Rdb7.

5.0.32 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.

5.0.33 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.

5.0.34 Interactive SQL Command Line Editor Rejects Eight-Bit Characters

Digital UNIX Systems

The interactive SQL command line editor on Digital UNIX can interfere with entering eight-bit characters from the command line. The command line editor assumes that a character with the eighth bit set will invoke an editing function. If the command line editor is enabled and a character with the eighth bit set is entered from the command line, the character will not be inserted on the command line. If the character has a corresponding editor function, the function will be invoked; otherwise, the character is considered invalid and rejected.

There are two ways to enter eight-bit characters from the SQL command line: either disable the command line editor or use the command line editor character quoting function to enter each eight-bit character. To disable the command line editor, set the configuration parameter RDB_NOLINEDIT in the configuration file. For example:

```
! Disable the interactive SQL command line editor.
RDB_NOLINEDIT   ON
```

To place quotation marks around a character using the command line editor, type Ctrl/V before each character to be place in quotation marks.

5.0.35 Restriction Added for CREATE STORAGE MAP on Table with Data

Oracle Rdb7 added support that allows a storage map to be added to an existing table which contains data. The restrictions listed for Oracle Rdb7 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 Rdb7 adds the additional restriction that the storage map may not include a WITH LIMIT clause for the storage area. The following example shows the reported error:

```
SQL> CREATE TABLE MAP_TEST1 (A INTEGER, B CHAR(10));
SQL> CREATE INDEX MAP_TEST1_INDEX ON MAP_TEST1 (A);
SQL> INSERT INTO MAP_TEST1 (A, B) VALUES (3, 'Third');
1 row inserted
SQL> CREATE STORAGE MAP MAP_TEST1_MAP FOR MAP_TEST1
cont> STORE USING (A) IN RDB$SYSTEM
cont>     WITH LIMIT OF (10);  -- can't use WITH LIMIT clause
%RDB-E-NO_META_UPDATE, metadata update failed
-RDMS-F-RELNOTEMPTY, table "MAP_TEST1" has data in it
-RDMS-E-NOCMPLXMAP, can not use complex map for non-empty table
```

5.0.36 ALTER DOMAIN...DROP DEFAULT Reports DEFVALUNS Error

Bug 456867

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

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

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

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

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

5.0.37 Oracle Rdb7 Workload Collection Can Stop Hot Standby Replication

If you are replicating your Oracle Rdb7 database using the Oracle Hot Standby option, you must not use the workload collection option. By default, workload collection is disabled. However, if you enabled workload collection, you must disable it on the master database prior to performing a backup operation on that master database if it will be used to create the standby database for replication purposes. If you do not disable workload collection, it could write workload information to the standby database and prevent replication operations from occurring.

The workaround included at the end of this section describes how to disable workload collection on the master database and allow the Hot Standby software to propagate the change to the standby database automatically during replication operations.

Background Information

By default, workload collection and cardinality collection are automatically disabled when Hot Standby replication operations are occurring on the standby database. However, if replication stops (even for a brief network failure), Oracle Rdb7 potentially can start a read/write transaction on the standby database to write workload collection information. Then, because the standby database is no longer synchronized transactionally with the master database, replication operations cannot restart.

Note

The Oracle Rdb7 optimizer can update workload collection information in the RDB$WORKLOAD system table even though the standby database is opened exclusively for read-only queries. A read/write transaction is started during the 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);
```

5.0.38 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.

5.0.39 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.

5.0.40 Restriction when Adding Storage Areas with Users Attached to Database

If you try to interactively add a new storage area where the page size is less than the existing page size and the database has been manually opened or users are active, the add operation fails with the following error:

```
%RDB-F-SYS_REQUEST, error from system services request
-RDMS-F-FILACCERR, error opening database root DKA0:[RDB]TEST.RDB;1
-SYSTEM-W-ACCONFLICT, file access conflict
```

You can make this change only when no users are attached to the database and, if the database is set to OPEN IS MANUAL, the database is closed. Several internal Oracle Rdb data structures are based on the minimum page size, and these structures cannot be resized if users are attached to the database.

Furthermore, because this particular change is not recorded in the AIJ 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.

5.0.41 Restriction on Tape Usage for Digital UNIX V3.2

Digital UNIX Systems

You can experience a problem where you are unable to use multiple tapes with the Oracle RMU Backup command with Digital UNIX V3.2. Every attempt to recover fails. If this happens and device errors are logged in the system error log, it is possible that the operation succeeded, but the device open reference count is zeroed out. This means that any attempt to use the drive by the process holding the open file descriptor will fail with EINVAL status but another process will be able to open and use the drive even while the first process has it opened.

There is no workaround for this problem. This problem with the magtape driver will be corrected in a future release of Digital UNIX.

5.0.42 Support for Single-File Databases to be Dropped in a Future Release

Oracle Rdb currently supports both single-file and multifile databases on both OpenVMS and Digital UNIX. However, single-file databases will not be supported in a future release of Oracle 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.

5.0.43 DECdtm Log Stalls

Resource managers using the DECdtm services can sometimes suddenly stop being able to commit transactions. If Oracle Rdb7 is installed and transactions are being run, an RMU Show command on the affected database will show transactions as being "stalled, waiting to commit".

Refer to the DECdtm documentation and release notes for information on symptoms, fixes, and workarounds for this problem. One workaround, for OpenVMS V5.5-x, is provided here.

On the affected node while the log stall is in progress, type the following command from a privileged account:

```
$ MCR LMCP SET NOTIMEZONE
```

This should force the log to restart.

This stall occurs only when a particular bit in a pointer field becomes set. To see the value of the pointer field, enter the following command from a privileged account (where <nodename> is the SCS node name of the node in question).

```
$ MCR LMCP DUMP/ACTIVE/NOFORM SYSTEM$<nodename>
```

This command displays output similar to the following:

```
Dump of transaction log SYS$COMMON:[SYSEXE]SYSTEM$<nodename>.LM$JOURNAL;1
End of file block 4002 / Allocated 4002
Log Version 1.0
Transaction log UID:   29551FC0-CBB7-11CC-8001-AA000400B7A5
Penultimate Checkpoint: 000013FD4479 0079
Last Checkpoint:        000013FDFC84 0084

Total of 2 transactions active, 0 prepared and 2 committed.
```

The stall will occur when bit 31 of the checkpoint address becomes set, as this excerpt from the previous example shows:

```
        Last Checkpoint:        000013FDFC84 0084
                                    ^
                                    |
```

When the number indicated in the example becomes 8, the log will stall. Check this number and observe how quickly it grows. When it is at 7FFF, frequently use the following command:

```
$ MCR LMCP SHOW LOG /CURRENT
```

If this command shows a stall in progress, use the workaround to restart the log.

See your Compaq Computer Corporation representative for information about patches to DECdtm.

5.0.44 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 Rdb7 operations that require the two-phase commit protocol. The two-phase commit protocol guarantees that if one operation in a distributed transaction cannot be completed, none of the operations is completed.

If you have DECnet/OSI installed on a system running OpenVMS VAX Version 6.1 or higher or OpenVMS Alpha 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*.

5.0.45 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.

5.0.46 Oracle Rdb7 Network Link Failure Does Not Allow DISCONNECT to Clean Up Transactions

If a program attaches to a database on a remote node and it loses the connection before the COMMIT statement is issued, there is nothing you can do except exit the program and start again.

The problem occurs when a program is connected to a remote database and updates the database, but then just before it commits, the network fails. When the commit executes, SQL shows, as it normally should, that the program has lost the link. Assume that the user waits for a minute or two, then tries the transaction again. The problem is that when the start transaction is issued for the second time, it fails because old information still exists about the previous failed transaction. This occurs even if the user issues a DISCONNECT statement (in Release 4.1 and earlier, a FINISH statement), which also fails with an RDB-E-IO_ERROR error message.

5.0.47 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.

5.0.48 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.

5.0.49 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.)

5.0.50 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.

5.0.51 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

5.0.52 Side Effect when Calling Stored Routines

When calling a stored routine, you must not use the same routine to calculate argument values by a stored function. For example, if the routine being called is also called by a stored function during the calculation of an argument value, passed arguments to the routine may be incorrect.

The following example shows a stored procedure P being called during the calculation of the arguments for another invocation of the stored procedure P:

```
SQL> CREATE MODULE M
cont>     LANG SQL
cont>
cont>     PROCEDURE P (IN :A INTEGER, IN :B INTEGER, OUT :C INTEGER);
cont>     BEGIN
cont>     SET :C = :A + :B;
cont>     END;
cont>
cont>     FUNCTION F () RETURNS INTEGER
cont>     COMMENT IS 'expect F to always return 2';
cont>     BEGIN
cont>     DECLARE :B INTEGER;
cont>     CALL P (1, 1, :B);
cont>     TRACE 'RETURNING ', :B;
cont>     RETURN :B;
cont>     END;
cont> END MODULE;
SQL>
SQL> SET FLAGS 'TRACE';
SQL> BEGIN
cont> DECLARE :CC INTEGER;
cont> CALL P (2, F(), :CC);
cont> TRACE 'Expected 4, got ', :CC;
cont> END;
~Xt: returning 2
~Xt: Expected 4, got 3
```

The result as shown above is incorrect. The routine argument values are written to the called routine's parameter area before complex expression values are calculated. These calculations may (as in the example) overwrite previously copied data.

The workaround is to assign the argument expression (in this example calling the stored function F) to a temporary variable and pass this variable as the input for the routine. The following example shows the workaround:

```
SQL> BEGIN
cont> DECLARE :BB, :CC INTEGER;
cont> SET :BB = F();
cont> CALL P (2, :BB, :CC);
cont> TRACE 'Expected 4, got ', :CC;
cont> END;
~Xt: returning 2
~Xt: Expected 4, got 4
```

This problem will be corrected in a future version of Oracle Rdb7.

5.0.53 Nested Correlated Subquery Outer References Incorrect

This problem was corrected in Oracle Rdb7 Release 7.0.0.2. An updated release note stating that this was fixed was inadvertently left out of all the following sets of release notes. Please note that this issue is now corrected. Outer references from aggregation subqueries contained within nested queries could receive incorrect values, causing the overall query to return incorrect results. The general symptom for an outer query that returned rows 1 to n was that the inner aggregation query would operate with the n^{th} - 1 row data (usually NULL for row 1) when it should have been using the n^{th} row data.

This problem has existed in various forms for all previous versions of Oracle Rdb7, but only appears in Release 6.1 and later when the inner of the nested queries contains an UPDATE statement.

The following example demonstrates the problem:

```
SQL> ATTACH 'FILENAME SHIPPING';
SQL> SELECT * FROM MANIFEST WHERE VOYAGE_NUM = 4904 OR
cont>                            VOYAGE_NUM = 4909;
  VOYAGE_NUM      EXP_NUM   MATERIAL             TONNAGE
        4904          311   CEDAR                   1200
        4904          311   FIR                      690
        4909          291   IRON ORE                3000
        4909          350   BAUXITE                 1100
        4909          350   COPPER                  1200
        4909          355   MANGANESE                550
        4909          355   TIN                      500
7 rows selected

SQL> BEGIN
cont> FOR :A AS EACH ROW OF
cont>  SELECT * FROM VOYAGE V WHERE V.SHIP_NAME = 'SANDRA C.' OR
cont>                              V.SHIP_NAME = 'DAFFODIL' DO
cont>   FOR :B AS EACH ROW OF TABLE CURSOR MODCUR1 FOR
cont>    SELECT * FROM  MANIFEST M WHERE M.VOYAGE_NUM = :A.VOYAGE_NUM DO
cont>     UPDATE MANIFEST
cont>      SET TONNAGE = (SELECT (AVG (M1.EXP_NUM) *3) FROM MANIFEST M1
cont>                    WHERE M1.VOYAGE_NUM = :A.VOYAGE_NUM)
cont>      WHERE CURRENT OF MODCUR1;
cont>   END FOR;
cont> END FOR;
cont> END;
SQL> SELECT * FROM MANIFEST WHERE VOYAGE_NUM = 4904 OR
cont>                             VOYAGE_NUM = 4909;
  VOYAGE_NUM      EXP_NUM   MATERIAL             TONNAGE
        4904          311   CEDAR                   NULL
        4904          311   FIR                     NULL
        4909          291   IRON ORE                 933
        4909          350   BAUXITE                  933
        4909          350   COPPER                   933
        4909          355   MANGANESE                933
        4909          355   TIN                      933
7 rows selected
```

The correct value for TONNAGE on both rows for VOYAGE_NUM 4904 (outer query row 1) is AVG (311+311)*3=933. However, Oracle Rdb7 calculates it as AVG (NULL+NULL)*3=NULL. In addition, the TONNAGE value for VOYAGE_NUM 4909 (outer query row 2) is actually the TONNAGE value for outer query row 1.

A workaround is to declare a variable of the same type as the outer reference data item, assign the outer reference data into the variable before the inner query that contains the correlated aggregation subquery, and reference the variable in the aggregation subquery. Keep in mind the restriction on the use of local variables in FOR cursor loops.

For example:

```
SQL> DECLARE :VN INTEGER;
SQL> BEGIN
cont> FOR :A AS EACH ROW OF
cont>  SELECT * FROM VOYAGE V WHERE V.SHIP_NAME = 'SANDRA C.' DO
cont>   SET :VN = :A.VOYAGE_NUM;
cont>   FOR :B AS EACH ROW OF TABLE CURSOR MODCUR1 FOR
cont>    SELECT * FROM MANIFEST M WHERE M.VOYAGE_NUM = :A.VOYAGE_NUM DO
cont>     UPDATE MANIFEST
cont>      SET TONNAGE = (SELECT (AVG (M1.EXP_NUM) *3) FROM MANIFEST M1
cont>                     WHERE M1.VOYAGE_NUM = :VN)
cont>      WHERE CURRENT OF MODCUR1;
cont>   END FOR;
cont> END FOR;
cont> END;
SQL> SELECT * FROM MANIFEST WHERE VOYAGE_NUM = 4904;
  VOYAGE_NUM       EXP_NUM   MATERIAL              TONNAGE
        4904           311   CEDAR                     933
        4904           311   FIR                       933
```

This problem was corrected in Oracle Rdb7 Release 7.0.0.2. An updated release note stating that this was fixed was inadvertently left out of all the following sets of release notes. Please note that this issue is now corrected.

5.0.54 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.

5.0.55 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.

5.0.56 SQL Pascal Precompiler Processes ARRAY OF RECORD Declarations Incorrectly

The Pascal precompiler for SQL gives an incorrect %SQL-I-UNMATEND error when it parses a declaration of an array of records. The precompiler does not associate the END statement with the record definition, and the resulting confusion in host variable scoping causes a fatal error.

A workaround for the problem is to declare the record as a type and then define your array of that type. For example:

```
    main.spa:

        program main (input,output);

        type
        exec sql include 'bad_def.pin';    !gives error
        exec sql include 'good_def.pin';   !ok
        var
            a : char;

        begin
        end.
----------------------------------------------------------------
    bad_def.pin

    x_record = record
    y  : char;
    variable_a:  array [1..50] of record
                a_fld1 : char;
                b_fld2  : record;
                            t : record
                                      v : integer;
                            end;
                end;
        end;
    end;
 ----------------------------------------------------------------

    good_def.pin

good_rec = record
        a_fld1 : char;
        b_fld2 : record
                t : record
                        v: integer;
                end;
        end;
end;

    x_record = record
        y  : char
        variable_a : array [1..50] of good_rec;
    end;
```

5.0.57 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.

5.0.58 Oracle RMU Commands Pause During Tape Rewind

Digital UNIX Systems

For Oracle Rdb Release 6.1 or higher on Digital UNIX, the Oracle RMU Backup and Restore commands pause under certain conditions.

If multiple tape drives are used for RMU Backup or RMU Restore commands and a tape needs to rewind, the Oracle RMU command pauses until the rewind is complete. This is different from behavior on OpenVMS systems where the command continues to write to tape drives that are not rewinding.

There is no workaround for this problem.

5.0.59 TA90 and TA92 Tape Drives Are Not Supported on Digital UNIX

Digital UNIX Systems

When rewinding or unloading tapes using either TA90 and TA92 drives, Digital UNIX intermittently returns an EIO error causing the Oracle RMU operation to abort. This problem occurs most often when Oracle RMU accesses multiple tape drives in parallel. However, the problem occurs even with single-tape drive access.

As a result of this problem, Oracle Rdb on Digital UNIX supports neither TA90 nor TA92 tape drives.

5.1 Oracle CDD/Repository Restrictions

This section describes known problems and restrictions in Oracle CDD/Repository Release 7.0 and earlier.

5.1.1 Oracle CDD/Repository Compatibility with Oracle Rdb Features

Some Oracle Rdb features are not fully supported by all versions of Oracle CDD/Repository. Table 5–1 shows which versions of Oracle CDD/Repository support Oracle Rdb features and the extent of support.

In Table 5–1, repository support for Oracle Rdb7 features can vary as follows:

- Explicit support—The repository recognizes and integrates the feature, and you can use the repository to manipulate the item.
- Implicit support—The repository recognizes and integrates the feature, but you cannot use any repository interface to manipulate the item.
- Pass-through support—The repository does not recognize or integrate the feature, but allows the Oracle Rdb7 operation to complete without aborting or overwriting metadata. With pass-through support, a CDD-I-MBLRSYNINFO informational message may be returned.

Table 5–1 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	Explict
Date arithmetic	4.1	6.1	Pass-through
DATE ANSI, TIME, TIMESTAMP, and INTERVAL data types	4.1	6.1	Explicit
Delimited identifiers	4.2	6.1[1]	Explicit
External functions	6.0	6.1	Pass-through
External procedures	7.0	6.1	Pass-through
EXTRACT, CHAR_LENGTH, and OCTET_LENGTH functions	4.1	6.1	Explicit
GRANT/REVOKE privileges	4.0	5.0 accepts but does not store information	Pass-through
Indexes	1.0	5.2	Explicit
INTEGRATE DOMAIN	6.1	6.1	Explicit
INTEGRATE TABLE	6.1	6.1	Explicit
Logical area thresholds for storage maps and indexes	4.1	5.2	Pass-through
Multinational character set	3.1	4.0	Explicit
Multiversion environment (multiple Rdb versions)	4.1	5.1	Explicit
NULL keyword	2.2	7.0	Explicit
Oracle7 compatibility functions, such as CONCAT, CONVERT, DECODE, and SYSDATE	7.0	7.0	Explicit
Outer joins, derived tables	6.0	7.0	Pass-through
Query outlines	6.0	6.1	Pass-through
Storage map definitions correctly restored	3.0	5.1	Explicit

[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.

(continued on next page)

Table 5–1 (Cont.) Oracle CDD/Repository Compatibility for Oracle Rdb Features

Oracle Rdb Feature	Minimum Release of Oracle Rdb	Minimum Release of Oracle CDD/Repository	Support
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

[2]Multivendor Integration Architecture (MIA) features include the CHAR_LENGTH clause and the TRANSLATE function.

5.1.2 Multischema Databases and CDD/Repository

You cannot use multischema databases with CDD/Repository and Oracle Rdb release 7.0 and earlier. This problem will be corrected in a future release of Oracle Rdb.

5.1.3 Interaction of Oracle CDD/Repository Release 5.1 and Oracle RMU Privileges Access Control Lists

Oracle Rdb provides special Oracle RMU privileges that use the unused portion of the OpenVMS access control list (ACL) to manage access to Oracle RMU operations.

You can use the RMU Set Privilege and RMU Show Privilege commands to set and show the Oracle RMU privileges. The DCL SHOW ACL and DIRECTORY/ACL commands also show the added access control information; however, these tools cannot translate the names defined by Oracle Rdb.

Note

The RMU Convert command propagates the database internal ACL to the root file for access control entries (ACEs) that possess the SECURITY and DBADM (ADMINISTRATOR) privileges.

Oracle CDD/Repository protects its repository (dictionary) by placing the CDD$SYSTEM rights identifier on each file created within the anchor directory. CDD$SYSTEM is a special, reserved rights identifier created by Oracle CDD/Repository.

When Oracle CDD/Repository executes the DEFINE REPOSITORY command, it adds (or augments) an OpenVMS default ACL to the anchor directory. Typically, this ACL allows access to the repository files for CDD$SYSTEM and denies access to everyone else. All files created in the anchor directory inherit this default ACL, including the repository database.

Unfortunately, there is an interaction between the default ACL placed on the repository database by Oracle CDD/Repository and the Oracle RMU privileges ACL processing.

Within the ACL on the repository database, the default access control entries (ACEs) that were inherited from the anchor directory will precede the ACEs added by RMU Restore. As a result, the CDD$SYSTEM identifier will not have any Oracle RMU privileges granted to it. Without these privileges, if the user does not have the OpenVMS SYSPRV privilege enabled, Oracle RMU operations, such as Convert and Restore, will not be allowed on the repository database.

The following problems may be observed by users who do not have the SYSPRV privilege enabled:

- While executing a CDO DEFINE REPOSITORY or DEFINE DICTIONARY command:
  - If the CDD$TEMPLATEDB backup (.rbf) file was created by a previous version of Oracle Rdb7, the automatic RMU Convert operation that will be carried out on the .rbf file will fail because SYSPRV privilege is required.
  - If the CDD$TEMPLATEDB backup (.rbf) file was created by the current version of Oracle Rdb7, the restore of the repository database will fail because the default ACEs that already existed on the repository file that was backed up will take precedence, preventing RMU$CONVERT and RMU$RESTORE privileges from being granted to CDD$SYSTEM or the user.
  - If no CDD$TEMPLATEDB is available, the repository database will be created without a template, inheriting the default ACL from the parent directory. The ACE containing all the required Oracle RMU privileges will be added to the end of the ACL; however, the preexisting default ACEs will prevent any Oracle RMU privilege from being granted.
- You must use the RMU Convert command to upgrade the database disk format to Oracle Rdb7 after installing Release 7.0. This operation requires the SYSPRV privilege.

  During the conversion, RMU Convert adds the ACE containing the Oracle RMU privileges at the end of the ACL. Because the repository database already has the default Oracle CDD/Repository ACL associated with it, the Oracle CDD/Repository ACL will take precedence, preventing the granting of the Oracle RMU privileges.
- During a CDO MOVE REPOSITORY command, the Oracle RMU privilege checking may prevent the move, as the RMU$COPY privilege has not been granted on the repository database.
- When you execute the CDD template builder CDD_BUILD_TEMPLATE, the step involving RMU Backup privilege has not been granted.

Oracle CDD/Repository Releases 5.2 and higher correct this problem. A version of the Oracle CDD/Repository software that corrects this problem and allows new repositories to be created using Oracle Rdb7 is provided on the Oracle Rdb7 kit for use on OpenVMS VAX systems. See Section 5.1.3.1 for details.

5.1.3.1 Installing the Corrected CDDSHR Images

OpenVMS VAX Systems

Note

The following procedure must be carried out if you have installed or plan to install Oracle Rdb7 and have already installed CDD/Repository Release 5.1 software on your system.

Due to the enhanced security checking associated with Oracle RMU commands in Oracle Rdb on OpenVMS VAX, existing CDDSHR images for CDD/Repository Release 5.1 must be upgraded to ensure that the correct Oracle RMU privileges are applied to newly created or copied repository databases.

Included in the Oracle Rdb7 for OpenVMS VAX distribution kit is a CDD upgraded image kit, called CDDRDB042, that must be installed after you have installed the Oracle Rdb7 for OpenVMS VAX kit.

This upgrade kit should be installed by using VMSINSTAL. It automatically checks which version of CDDSHR you have installed and replaces the existing CDDSHR.EXE with the corrected image file. The existing CDDSHR.EXE will be renamed SYS$LIBRARY:OLD_CDDSHR.EXE.

The upgrade installation will also place a new CDD_BUILD_TEMPLATE.COM procedure in SYS$LIBRARY for use with CDD/Repository V5.1.

Note

If you upgrade your repository to CDD/Repository V5.1 after you install Oracle Rdb7 V7.0, you must install the corrected CDDSHR image again to ensure that the correct CDDSHR images have been made available.

The CDD/Repository upgrade kit determines which version of CDD/Repository is installed and replaces the existing CDDSHR.EXE with the appropriate version of the corrected image.

5.1.3.2 CDD Conversion Procedure

OpenVMS VAX Systems

Oracle Rdb7 provides RDB$CONVERT_CDD$DATABASE.COM, a command procedure that both corrects the anchor directory ACL and performs the RMU Convert operation. The command procedure is located in SYS$LIBRARY.

Note

You must have SYSPRV enabled before you execute the procedure RDB$CONVERT_CDD$DATABASE.COM because the procedure performs an RMU Convert operation.

Use the procedure RDB$CONVERT_CDD$DATABASE.COM to process the anchor directory and update the ACLs for both the directory and, if available, the repository database.

This procedure accepts one parameter: the name of the anchor directory that contains, or will contain, the repository files. For example:

```
$ @SYS$LIBRARY:DECRDB$CONVERT_CDD$DATABASE [PROJECT.CDD_REP]
```

If many repositories exist on a system, you may want to create a DCL command procedure to locate them, set the Oracle RMU privileges ACL, and convert the databases. Use DCL commands similar to the following:

```
$ LOOP:
$       REP_SPEC = F$SEARCH("[000000...]CDD$DATABASE.RDB")
$       IF REP_SPEC .NES. ""
$       THEN
$           @SYS$LIBRARY:DECRDB$CONVERT_CDD$DATABASE    -
                'F$PARSE(REP_SPEC,,,"DIRECTORY")'
$           GOTO LOOP
$       ENDIF
```

6
Enhancements

This chapter describes the enhancements that are introduced in Oracle Rdb7 Release 7.0.5.

6.1 Enhancements Provided in Oracle Rdb7 Release 7.0.5

6.1.1 SHOW STATISTIC "Checkpoint Analysis" Screen

The RMU Show Statistic Utility “Online Analysis” facility has been enhanced. The “Checkpoint Analysis” screen performs basic review and analysis of the database checkpoint and process recovery information. The purpose of this screen is to identify processes whose recovery may impact database throughput or availability.

The “Checkpoint Analysis” screen is available even when the “AIJ Fast Commit” feature is not enabled. However, some of the analysis may not be performed in this case.

The following is an example of the “Checkpoint Analysis” screen display:

```
Node: ALPHA3 (1/1/1)    Oracle Rdb X7.1-00 Perf. Monitor 13-JAN-2000 08:05:49.34
Rate: 1.00 Second             Checkpoint Analysis        Elapsed: 5 21:17:06.33
Page: 1 of 1      KODA_TEST:[R_ANDERSON.TCS_MASTER]TCS.RDB;1      Mode: Global
--------------------------------------------------------------------------------

Process 3C82F330:1 checkpoint 15:2 lags behind current AIJ sequence 18
Process 3C82F330:1 checkpoint 15:2 exceeds 512 block threshold
Process 3C82F330:1 process recovery duration 123 seconds exceeds 10 threshold
Process 3C82F330:1 database freeze duration 123 seconds exceeds 15 threshold
Process 3C82E731:1 checkpoint 17:6586 lags behind current AIJ sequence 18
Process 3C82E731:1 checkpoint 17:6586 exceeds 512 block threshold
Process 3C82E731:1 database freeze duration 6 seconds exceeds 15 threshold
Process 3C82FD33:35 checkpoint 18:2216 exceeds 512 block threshold
Process 3C827752:1 checkpoint 18:2217 exceeds 512 block threshold
Process 3C82DF55:1 checkpoint 18:2142 exceeds 512 block threshold
Process 3C830948:1 checkpoint 18:2218 exceeds 512 block threshold

--------------------------------------------------------------------------------
Config Exit Help Menu Set_rate Write !
```

The “Checkpoint Analysis” screen performs the following analysis operations:

- **Checkpoint Stale.** This analysis determines whether or not the process checkpoint occurs within the current AIJ journal, which is always desireable.

  This analysis results in the following message being displayed:

  ```
  Process 3C82F330:1 checkpoint 15:2 lags behind current AIJ sequence 18
  ```

- **Checkpoint Old.** This analysis determines whether or not the checkpoint size exceeds a user-specified threshold, expressed in AIJ blocks. The number of AIJ blocks constitutes a physical process recovery duration, but also impacts other components, such as AIJ backup and Row Cache.

The default checkpoint block count threshold is 512 blocks. The default threshold can be modified in the following manner:

- The logical RDM$BIND_STATS_CHECKPOINT_BLOCK_COUNT can be defined to specify a different threshold at utility startup.
- The configuration variable CHECKPOINT_BLOCK_COUNT can be defined to specify a different threshold in the configuration file.
- The threshold can be modified at run-time using the "Config" on-screen menu option, by selecting the "Checkpoint block count" option.

This analysis results in the following message being displayed:

```
Process 3C82F330:1 checkpoint 15:2 exceeds 512 block threshold
```

- **RUJ File Size.** This analysis determines whether or not the process RUJ file size exceeds a user-specified threshold, expressed in blocks. The number of RUJ blocks constitutes a transaction recovery duration.

  The default RUJ file size threshold is 256 blocks. The default threshold can be modified in the following manner:

  - The logical RDM$BIND_STATS_RUJ_FILE_SIZE can be defined to specify a different threshold at utility startup.
  - The configuration variable RUJ_FILE_SIZE can be defined to specify a different threshold in the configuration file.
  - The threshold can be modified at run-time using the "Config" on-screen menu option, by selecting the "RUJ file size" option.

  This analysis results in the following message being displayed:

  ```
  Process 3C82C943:1 RUJ size 30 block exceeds 25 block threshold
  ```

- **Transaction Rollback Duration.** This analysis determines whether or not the transaction rollback duration exceeds a user-defined threshold, expressed in seconds.

  The default transaction rollback threshold is 5 seconds. The default threshold can be modified in the following manner:

  - The logical RDM$BIND_TX_UNDO_DURATION can be defined to specify a different threshold at utility startup.
  - The configuration variable TX_UNDO_DURATION can be defined to specify a different threshold in the configuration file.
  - The threshold can be modified at run-time using the "Config" on-screen menu option, by selecting the "Transaction rollback duration" option.

  This analysis results in the following message being displayed:

  ```
  Process 3C82F330:1 Transaction rollback duration 13 seconds exceeds 10 second
  threshold
  ```

- **Process Recovery Duration.** This analysis determines whether or not the recovery of a process failure (transaction REDO) exceeds a user-defined threshold, expressed in seconds.

  The default process recovery threshold is 10 seconds. The default threshold can be modified in the following manner:

  - The logical RDM$BIND_TX_REDO_DURATION can be defined to specify a different threshold at utility startup.

  - The configuration variable TX_REDO_DURATION can be defined to specify a different threshold in the configuration file.
  - The threshold can be modified at run-time using the “Config” on-screen menu option, by selecting the “Process recovery duration” option.

  This analysis results in the following message being displayed:

  ```
  Process 3C82F330:1 process recovery duration 123 seconds exceeds 10 second
  threshold
  ```

- **Database freeze duration.** This analysis determines whether or not the entire database freeze duration exceeds a user-defined threshold, expressed in seconds. The database freeze duration includes both transaction rollback, process recovery (transaction REDO) and DBR processing.

  The default database freeze threshold is 15 seconds. The default threshold can be modified in the following manner:

  - The logical RDM$BIND_DBR_FREEZE_DURATION can be defined to specify a different threshold at utility startup.
  - The configuration variable TX_DBR_FREEZE_DURATION can be defined to specify a different threshold in the configuration file.
  - The threshold can be modified at run-time using the “Config” on-screen menu option, by selecting the “Database freeze duration” option.

  This analysis results in the following message being displayed:

  ```
  Process 3C82F330:1 Database freeze duration 123 seconds exceeds 5 second
  threshold
  ```

The “Checkpoint Analysis” screen information displays run-time information and is not recorded in the binary output file. Consequently, the screen is also not available during replay of a binary input file.

The information displayed represents information for the current node only, even when cluster-wide statistics collection is enabled.

6.1.2 RMU Show Statistic "Online Analysis Logfile" Facility

The RMU Show Statistic utility has been enhanced to provide an “Online Analysis” log file. The online analysis log file provides hard copy of all of the analysis performed by all of the “Online Analysis” facility screens, without having to actually display any of the screens.

The “Online Analysis” log file is enabled using two different methods:

1. The configuration variable ONLINE_ANALYSIS_LOG identifies the name of the log file to contain the online analysis information.
2. At run-time, the online analysis log file can be started and stopped using the Tools menu, obtained via the “!” shortcut. Selecting the “Start online analysis logging” option will create the log file, and selecting the “Stop online analysis logging” will terminate the log file.

Note

There is no command qualifier to directly enable the online analysis log file; the configuration file should be used instead via the /CONFIG command qualifier.

The following shows an example of the online analysis log file contents:

```
        Oracle Rdb X7.1-00 Performance Monitor Online Analysis Log
        Database KODA_TEST:[R_ANDERSON.TCS_MASTER]TCS.RDB;1
        Online Analysis Log created 15-JAN-2000 07:36:26.69
07:36:34.56 95th %ile transaction duration: 34.4 seconds
07:36:34.56 95th %ile read/write transaction duration: 31.9 seconds
07:36:34.56 95th %ile read-only transaction duration: 503.9 seconds
07:36:34.56 Log server is Automatic
07:36:34.56 AIJ TCS1 device DPA48: same as storage area
07:36:34.56 AIJ TCS2 device DPA48: same as storage area
07:36:34.56 AIJ TCS3 device DPA48: same as storage area
07:36:34.56 AIJ TCS4 device DPA48: same as storage area
07:36:34.56 AIJ TCS5 device DPA48: same as storage area
07:36:34.56 ARB pool exhausted 56 times
07:36:34.56 12.2% synchronous RUJ I/O above 10.0% threshold
07:36:34.56 9.5% RUJ extends above 2.0% threshold
07:36:34.56 Process 3C830F2A:1 checkpoint 11:1075 exceeds 512 block threshold
07:36:34.56 Process 3C830F2A:1 process recovery duration 14 seconds exceeds 10
    second threshold
07:36:34.56 Process 3C83112B:1 checkpoint 10:4911 lags behind current AIJ
    sequence 11
07:36:34.56 Process 3C83112B:1 checkpoint 10:4911 exceeds 512 block threshold
07:36:34.56 Process 3C83112B:1 process recovery duration 21 seconds exceeds 10
    second threshold
07:36:34.56 Process 3C83112B:1 database freeze duration 21 seconds exceeds 15
    second threshold
07:36:34.56 Process 3C831732:31 checkpoint 11:14290 exceeds 512 block threshold
07:36:34.56 Process 3C831D3C:1 checkpoint 11:14213 exceeds 512 block threshold
07:36:34.56 92617.6 process recovery duration above 2.0 second threshold
07:36:34.56 Full database backup has not been performed since  7-JAN-2000
    13:46:55.60
07:36:34.56 17.7% page discard rate above 10.0% threshold (avg 0.1 I/Os)
07:36:34.56 100.0% SPAM page fetch rate above 80.0% total fetched threshold
07:36:34.56 217.6% SPAM page fetch rate above 20.0% record stored threshold
07:36:34.56 data TCS extended 1 time total 0 times
07:36:34.56 data TCS async write I/O stalls 1.2 exceeded average 0.3
07:36:34.56 data TCS sync write I/O stalls 15.7 exceeded average 3.0
07:36:42.49 Process 3C82DD5C:1 excessive deadlocks 12 on waiting for page
    10:2757 (PR)
07:36:42.49 258.8% duplicate btree fetch above 15.0% threshold
07:36:42.49 51.0% duplicate btree store above 15.0% threshold
07:36:42.49 34.6% duplicate hash btree fetch above 15.0% threshold
07:36:42.49 34.4% duplicate hash index store above 15.0% threshold
07:36:42.49 Row cache is not allowed
```

The online analysis is performed at the specified screen refresh rate. It is possible to generate a considerable number of entries in the online analysis log file. Therefore, it is recommended that the online analysis log file be used in a non-interactive batch job with a reasonable refresh rate of five, ten or thirty seconds.

6.1.3 New OPTIMIZE Clause DML Statements

Oracle Rdb7 Release 7.0.5 adds a new OPTIMIZE FOR SEQUENTIAL ACCESS clause to SELECT, DELETE and UPDATE statements which want to force the use of sequential access. This is particularly valuable for tables which use the strict partitioning functionality.

When a table's storage map has the attribute PARTITIONING IS NOT UPDATABLE, the mapping of data to a storage area is strictly enforced. This is known as **strict partitioning**. When queries on such tables use sequential access, the optimizer can eliminate partitions which do not match the query WHERE restriction rather than scan every partition.

The following example shows a query that deletes selected rows from a specific partition. This table also includes several indices which may be chosen by the optimizer. This new OPTIMIZE clause forces sequential access. In previous releases, a query outline would have to be created for this query. This new clause effectively creates this query outline on-the-fly.

```
SQL> delete from PARTS_LOG
cont> where parts_id between 10000 and 20000
cont>   and expiry_date < :purge_date
cont> optimize for sequential access;
```

Please note that all access performed by such queries will be sequential access. Care should be taken that the I/O being used is acceptable by comparing similar queries using index access.

6.1.4 New RMU /RECLAIM Command

Applications that specify the database attach attribute DBKEY SCOPE IS ATTACH can accumulate locked space and locked DBKEYs within the database. If one user is connected to the database in DBKEY SCOPE IS ATTACH mode, all users are forced to operate in this mode, even if they are are explicitly connected in TRANSACTION mode. That is, no one reuses dbkeys until the ATTACH session disconnects.

A new RMU /RECLAIM command has been added to allow database keys of deleted rows to be rapidly "cleaned up" in one or more storage areas. The RMU /RECLAIM command reads and updates all pages in a storage area. Where possible, locked lines and locked free space are “released” so that they will be available for later allocation.

The RMU /RECLAIM command runs on-line (does not require exclusive access). However, if there are any users connected to the database in DBKEY SCOPE IS ATTACH mode, the RMU /RECLAIM operation will have greatly reduced effect. In order to release all possible locked space, there should be no users attached to the database in DBKEY SCOPE IS ATTACH mode. Further, to allow database page locked space to be reclaimed, the database session that “owns” the locked space must be detached from the database. This can be accomplished by having each database attach disconnect and reconnect to the database.

Valid qualifiers for the “RMU /RECLAIM” command are:

- /AREA=(listofareas) to indicate the storage areas to be reclaimed. The default is to reclaim all storage areas.
- /LOG to display a log message at the completion of each storage area.

6.1.5 New RMU /SERVER RECORD_CACHE CHECKPOINT Command

A new “RMU /SERVER RECORD_CACHE CHECKPOINT” command has been added to allow a DBA to force the Record Cache Server (RCS) process to checkpoint all modified rows from cache back to the database. This command also accepts the optional qualifiers “/LOG” and “ /WAIT”.

6.1.6 RCS Cycles TID Value at Checkpoint Completion

When the Oracle Rdb7 Row Cache feature is enabled, the Row Cache Server (RCS) process will cycle through its transaction ID (TID) values as checkpoint or sweep operations that write modified data from the cache to the database complete. This cycling is intended to free locked rows on database pages from records that have been deleted so that the database keys and page space are available to other processes inserting records into the database.

6.1.7 New Option for the GET DIAGNOSTICS Statement/HOT_STANDBY_MODE

For Oracle Rdb7 Release 7.0.5, a new option has been added to the GET DIAGNOSTICS statement (this option was also available in Release 7.0.4 but the release note on it was mistakenly omitted).

- HOT_STANDBY_MODE

  This option returns a text string that indicates if this database is participating in a Hot Standby configuration as a master (returns 'MASTER'), or a standby (returns 'STANDBY'), or is not in such a configuration (returns 'NONE').

  The result data type is CHAR (31).

The following example uses the new option.

```
SQL> set flags 'trace';
SQL> declare :hsmode char(31);
SQL> begin
cont> get diagnostics :hsmode = HOT_STANDBY_MODE;
cont> trace :hsmode;
cont> end;
~Xt: NONE
SQL>
```

6.1.8 New Option for the GET DIAGNOSTICS Statement/TRANSACTION_CHANGE_ALLOWED

For Oracle Rdb7 Release 7.0.5, a new option has been added to the GET DIAGNOSTICS statement.

- TRANSACTION_CHANGE_ALLOWED

  There are many situations where the SQL language programmer would like to start or end a transaction but does not know if a transaction statement (SET TRANSACTION, COMMIT or ROLLBACK) is currently permitted. The transaction statements are not permitted in the following cases:

  - During a multi-database or global transaction. In this case the transaction must be coordinated by the client, not a server based procedure.
  - When a BEGIN ATOMIC compound statement is in the outer scope.
  - When a FOR cursor loop is active in an outer scope.

  This option allows the programmer to detect these restricted locations and conditionally execute a COMMIT, ROLLBACK or SET TRANSACTION as needed.

  The result data type is INTEGER. If transaction changes are permitted then a value one (1) will be assigned. Otherwise the result will be zero (0).

The following example shows one use of this new option. The called stored procedure ensures that changes to the transaction state are allowed before proceeding with a ROLLBACK.

```
SQL> create module M1
cont>     language SQL
cont>
cont>     procedure ROLLBACK_THE_CHANGE
cont>     comment is  'Perform a ROLLBACK only '
cont>     /           'if it is permitted';
cont>     begin
cont>     declare :txn_change integer;
cont>     get diagnostics
cont>         :txn_change = TRANSACTION_CHANGE_ALLOWED;
cont>     if :txn_change = 1
cont>     then
cont>         trace '...rolling back';
cont>         rollback;
cont>     else
cont>         trace '...skipping rollback';
cont>     end if;
cont>     end;
cont>
cont> end module;
SQL>
SQL> create table RT (a integer);
SQL> insert into RT (a) values (1);
1 row inserted
SQL> commit;
SQL>
SQL> set flags 'trace';
SQL>
SQL> begin
cont> call ROLLBACK_THE_CHANGE ();
cont> set transaction read only;
cont> for :x
cont>     as select * from RT
cont> do
cont>     call ROLLBACK_THE_CHANGE ();
cont>     trace :x.a;
cont> end for;
cont> end;
~Xt: ...rolling back
~Xt: ...skipping rollback
~Xt: 1
SQL>
```

6.1.9 New Hot Standby Logicals

Three new logicals have been added to the Hot Standby product:

- RDM$BIND_HOT_NETWORK_RETRY_COUNT

  This logical specifies the number of times the Hot Standby product should attempt to re-connect a disconnected network link. The default value is "1". There is no minimum nor maximum value (the value "0" means do not attempt to re-connect).

- RDM$BIND_HOT_NETWORK_RETRY_DELAY

  This logical specifies the number of seconds to wait before attempting to re-connect a disconnected network link, expressed in seconds. The default value is "1" second. There is no minimum nor maximum value (the value "0" means to immediately attempt to re-connect).

- RDM$BIND_LRS_BACKUP_AIJ

This logical specifies that the after-image journals on the standby database should be backed up, instead of merely being initialized, by the AIJ Backup Server (ABS). This logical accepts the following values: “0” indicates that the AIJ files should be initialized (this is the default); the value “1” indicates the AIJ files should be backed up (backup filespecs must have been previously configured).

6.2 Enhancements Provided in Oracle Rdb7 Release 7.0.4

6.2.1 Suggestion To Increase Field Size On RMU SHOW STATISTIC

The RMU/Show Statistic utility, in the menu under “Logical Area Information” sub-menu, “Logical Area Overview (Tables)” option, the “Logical Area Name” is limited to 20 characters. Customers frequently have table names that are larger than 20 characters, or they might have a tablename.areaname, and if this table is partitioned, some of their area files might have the same beginning part of the name with the end being different. It would be nice to have that 20 characters extend out further. Added per customer request.

The following example shows the current display:

```
Node: ALPHA3 (1/1/24)   Oracle Rdb X7.1-00 Perf. Monitor  2-NOV-1999 13:44:58.20
Rate: 1.00 Second        Logical Area Overview (Tables   Elapsed: 14 07:07:01.47
Page: 1 of 5       KODA_TEST:[R_ANDERSON.TCS_MASTER]TCS.RDB;1       Mode: Global
--------------------------------------------------------------------------------
Logical.Area.Name... record fetch record store record erase    discarded CurTot
RDB$RELATIONS.RDB$SY        24565            0            0            0
RDB$FIELD_VERSIONS.R       223904            0            0            0
RDB$INDICES.RDB$SYST        31495           15           23            0
RDB$INDEX_SEGMENTS.R        31064           45           69            0
RDB$FIELDS.RDB$SYSTE        27114            0            0            0
RDB$RELATION_FIELDS.        22520            0            0            0
RDB$DATABASE.RDB$SYS         1244            0            0            0
RDB$VIEW_RELATIONS.R            0            0            0            0
RDB$CONSTRAINT_RELAT            0            0            0            0
RDB$CONSTRAINTS.RDB$            0            0            0            0
RDB$STORAGE_MAPS.RDB         2056           15           23            0
RDB$STORAGE_MAP_AREA          524           15           23            0
RDB$INTERRELATIONS.R            0            0            0            0
RDB$COLLATIONS.RDB$S            0            0            0            0
RDB$TRIGGERS.RDB$SYS            0            0            0            0
RDB$RELATION_CONSTRA            0            0            0            0
RDB$RELATION_CONSTRA            0            0            0            0
--------------------------------------------------------------------------------
Config Exit Help Menu >next_page <prev_page Options Pause Reset Set_rate Write
```

There is no workaround to this problem.

This problem has been corrected in Oracle Rdb7 Release 7.0.4. By increasing the terminal display width, the RMU/Show Statistic utility will display a larger portion of the logical area name. For example, with the terminal width set to 90 columns, the above screen appears as follows:

```
Node: ALPHA3 (1/1/24)       Oracle Rdb X7.1-00 Perf. Monitor      2-NOV-1999 13:47:15.35
Rate: 1.00 Second            Logical Area Overview (Tables)      Elapsed: 14 07:09:18.62
Page: 1 of 5           KODA_TEST:[R_ANDERSON.TCS_MASTER]TCS.RDB;1          Mode: Global
------------------------------------------------------------------------------------------
Logical.Area.Name............. record fetch record store record erase   discarded CurTot
RDB$RELATIONS.RDB$SYSTEM             24565            0            0           0
RDB$FIELD_VERSIONS.RDB$SYSTEM       223904            0            0           0
RDB$INDICES.RDB$SYSTEM               31495           15           23           0
RDB$INDEX_SEGMENTS.RDB$SYSTEM        31064           45           69           0
RDB$FIELDS.RDB$SYSTEM                27114            0            0           0
RDB$RELATION_FIELDS.RDB$SYSTEM       22520            0            0           0
RDB$DATABASE.RDB$SYSTEM               1244            0            0           0
RDB$VIEW_RELATIONS.RDB$SYSTEM            0            0            0           0
RDB$CONSTRAINT_RELATIONS.RDB$S           0            0            0           0
RDB$CONSTRAINTS.RDB$SYSTEM               0            0            0           0
RDB$STORAGE_MAPS.RDB$SYSTEM           2056           15           23           0
RDB$STORAGE_MAP_AREAS.RDB$SYST         524           15           23           0
RDB$INTERRELATIONS.RDB$SYSTEM            0            0            0           0
RDB$COLLATIONS.RDB$SYSTEM                0            0            0           0
RDB$TRIGGERS.RDB$SYSTEM                  0            0            0           0
RDB$RELATION_CONSTRAINTS.RDB$S           0            0            0           0
RDB$RELATION_CONSTRAINT_FLDS.R           0            0            0           0
------------------------------------------------------------------------------------------
Config Exit Help Menu >next_page <prev_page Options Pause Reset Set_rate Write Zoom !
```

6.2.2 SHOW STATS "Logical Area Overview" Enhancements

Currently, the RMU Show Statistic Utility “Logical Area Overview” screen can only be sorted in alphabetical order. This is ideal for finding statistic information for a particular logical area, but is less than ideal when the screen is used for performance analysis.

The RMU Show Statistic Utility “Logical Area Overview” screen has been enhanced to provide the ability to sort the display based on any of the displayed column information. Since the user can configure the screen to display any statistic information in any column, this enhancement provides an extremely powerful tool for performance analysis.

Use the “Config” on-screen menu option to display the available sort options.

The following example shows a sample “Logical Area Overview” screen sorted on column one, records fetched:

```
Node: ALPHA3 (1/1/1)    Oracle Rdb X7.1-00 Perf. Monitor  5-NOV-1999 12:55:34.87
Rate: 0.50 Seconds      Logical Area Overview (Tables)   Elapsed: 2 03:00:48.99
Page: 1 of 4      KODA_TEST:[R_ANDERSON.TCS_MASTER]TCS.RDB;1      Mode: Global
--------------------------------------------------------------------------------
Logical.Area.Name... record fetch record store record erase    discarded CurTot
TRAN_SUMMARY_RECON2         43278        53187            0            0
TRAN_SUMMARY_RECON4         41732        53187            0            0
TRAN_SUMMARY_RECON6         35282        53187            0            0
TRAN_SUMMARY_RECON8         28196         8192            0            0
TRAN_SUMMARY_RECON12        16384        16384            0            0
TRAN_SUMMARY_RECON14         8192        16384            0            0
TRAN_SUMMARY_RECON16         8192        16384            0            0
LANE_MESSAGE_ID                 0            0            0            0
STATS_FILE_INDEX                0            0            0            0
SERIAL_KEY_INDEX                0            0            0            0
LANE                            0            0            0            0
XFER_CONTROL                    0            0            0            0
SCHEDULE_MASTER                 0            0            0            0
POOL_CARD                       0            0            0            0
EMPLOYEES                       0       196552            0            0
CSC_RECON                       0            0            0            0
SECURITY_MODULES                0            0            0            0
--------------------------------------------------------------------------------
Config Exit Help Menu >next_page <prev_page Options Pause Reset Set_rate Write
```

6.2.3 RCS Can Map All Caches at Database Open

By default, when the Oracle Rdb7 Row Cache feature is enabled, the first process to access a cached table or storage area will create and map the associated row cache(s). However, it is possible to cause the RCS process to create and map all defined row caches when the database is opened.

If the system-wide logical name RDM$BIND_RCS_INITIAL_MAP_ALL_CACHES is defined to the value "1" when the RCS process starts, the RCS process will create and map all defined row caches for the database.

The RCS process sorts the cache definitions for 32-bit address space usage from largest to smallest before the caches are created. This should help reduce memory fragmentation when using the SHARED MEMORY IS SYSTEM option.

6.2.4 Performance Enhancements When Number of Cluster Nodes is 1

Several performance enhancements have been made to Oracle Rdb7. The majority of these improvements are available for databases allowing access from only one node in a cluster (ie, the database is set to NUMBER OF CLUSTER NODES IS 1).

In particular, when the database is set to NUMBER OF CLUSTER NODES IS 1, various locks and root file write operations have been eliminated. Particularly in environments where this is a mix of read-only and read-write transactions, this can have a significant performance impact.

The logical name RDM$BIND_AWL_TSNBLK_LOCKING can be set to "1" to *avoid* several of these performance optimizations if desired.

Additionally, several internal data structures have been further aligned in memory for improved memory access patterns and shorter code sequences on Alpha processors.

6.2.5 New ROW LENGTH Default Calculated for CREATE CACHE

In prior versions of Oracle Rdb, when CREATE CACHE was used to define a logical cache for an existing table or index but the ROW LENGTH IS clause was omitted, the default length used was determined from the Area Inventory Page (AIP) entry for the logical area. Use RMU/DUMP/LAREA=RDB$AIP to see these length values. This default would sometimes be an inaccurate measure of the row or node size.

With Oracle Rdb7 Release 7.0.4, the defaulting has been changed so that the row length is derived from the Rdb metadata.

- When creating a logical cache, if a table already exists with the same name as the cache, then that table's current row length is calculated. This will account for any table changes which may have been made since the table was created, i.e. a column was added, dropped or altered in data type or size.

  In prior releases, the value used from the AIP may have been out-of-date and may have been too small (rows would not be cached) or too large (memory would be wasted).

  The default ROW LENGTH does not take into account the compression attributes of the table. Database administrators are encouraged to compare the default chosen with the actual rows on disk because row compression may allow a smaller row length to be used.

  Note

  If data in the row is not compressible then it is possible that the compression markers added to the row data will cause the length of the stored row to exceed the default ROW LENGTH. Please examine the stored data to see if this is a concern.

  If the table is vertically partitioned then this new default will represent the full row size, not the size of individual partitions. Oracle recommends that care be taken to calculate appropriate ROW LENGTH when caching vertically partitioned tables. In previous versions, the length used was for the first matching logical area which didn't necessarily provide a useful length.

  System tables do not explicitly use row compression but are stored in an internal abbreviated format. Therefore, Oracle does not recommend using the default ROW LENGTH for Oracle Rdb system tables but rather database administrators should calculate an appropriate value using the existing data in the system table.

- When creating a logical cache, if a SORTED index exists with the same name as the row cache, then the NODE SIZE specified by the CREATE or ALTER INDEX statement will be used for the ROW LENGTH. If none was used then the default size provided by Rdb will be used (typically this is 430 bytes).

  In prior releases the value used from the AIP for a SORTED index allowing duplicates was 215 bytes which was too small to cache the index nodes and may have only allowed the duplicate nodes to be cached.

  Note

  If an index and a table are given the same name, then Oracle Rdb will use the length from the table and not the index. In this case you must use an explicit ROW LENGTH clause to provide an acceptable length.

- In all other cases, the ROW LENGTH will default to 256. Oracle recommends that you calculate and specify an appropriate ROW LENGTH when creating physical caches or when creating logical caches for hashed index nodes.

These changes will have no affect on existing row cache definitions.

6.2.6 RMU /CHECKPOINT /WAIT /UNTIL

A new qualifier "/UNTIL=date-and-time" has been added to the "RMU /CHECKPOINT /WAIT" command. The UNTIL qualifier specifies the time at which the RMU /CHECKPOINT /WAIT command will stop waiting for the checkpoint and will return an error back to the user.

If you do not specify the UNTIL qualifier, the wait is indefinite.

6.2.7 RMU Extract Supports New AUDIT_COMMENT Option

Oracle Rdb7 Release 7.0.4 adds new functionality to RMU Extract. A new AUDIT_COMMENT option has been added that annotates the extracted objects with the creation and last alter timestamps as well as the username of the creator. The date/time values are displayed using the current settings of SYS$LANGUAGE and LIB$DT_FORMAT.

The default is /OPTION=NOAUDIT_COMMENT.

The following example shows an extract from the generated script when the SYS$LANGUAGE and LIB$DT_FORMAT 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 1998 13:01:31.20
-- Never altered
-- Created by RDB_EXECUTE
--
create domain ADDRESS_DATA_1
    CHAR (25);
    comment on domain ADDRESS_DATA_1 is
      ' Street name';
   .
   .
   .
```

6.2.8 Revised Oracle Rdb for OpenVMS Client Kit

The content of the Oracle Rdb for OpenVMS Client kit has been revised to reflect current software. This release of the client kit contains the following software:

- DBAPack V7.0.1 for the Windows NT on Intel, Windows 95 and Windows 98 platforms
- SQL/Services for Oracle Rdb Version 7.0-4
- SQL/Services Client for Compaq Tru64 UNIX Version 4.0
- SQL/Services Client for Sun Solaris

- Oracle Rdb ODBC Version 2.10.17 (32 bit version)
- Oracle Rdb ODBC for Mac OS
- Oracle Installer Version 3.3.1.2.4 (replaces Version 3.3.1.0.0)

The following software is no longer provided as part of the Oracle Rdb on OpenVMS Client kit.

- DBAPack for the Windows NT on DEC Alpha platform
- DBAPack for the Windows 3.1 platform
- Oracle Enterprise Manager Version 1.3.5
- Personal Oracle7 Version 7.0.3

7
LogMiner for Rdb

Oracle Rdb after-image journal (.aij) files contain a wealth of useful information about the history of transactions in a database. After-image journal files contain all of the data needed to perform database recovery. These files record every change made to data and metadata in the database. The LogMiner for Rdb feature provides an interface to the data record contents of Oracle Rdb after-image journal files. Data records that are added, updated, or deleted by committed transactions may be extracted (unloaded) from the .aij files in a format suitable for subsequent loading into another database or for use by user-written application programs.

Oracle Rdb after-image journaling protects the integrity of your data by recording all changes made by committed transactions to a database in a sequential log or journal file. Oracle Corporation recommends that you enable after-image journaling to record your database transaction activity between full backup operations as part of your database restore and recovery strategy. The after-image journal file is also used to enable several database performance enhancements (such as the fast commit, row cache, and hot standby features).

See the *Oracle Rdb7 Guide to Database Maintenance* for more information about setting up after-image journaling.

To use LogMiner for Rdb, follow these steps:

1. Enable the database for LogMiner operation using the RMU Set Logminer command. See Section 7.1 for additional information.
2. Back up the after-image journal file using the Quiet_Point qualifier to the RMU Backup command.
3. Extract changed records using the RMU Unload After_Journal command. See Section 7.2 for additional information.

7.1 RMU Set Logminer Command

Allows you to change the LogMiner state of a database.

Format

RMU/Set Logminer root-file-spec

Command Qualifiers	Defaults
/Disable	See description
/Enable	See description
/[No]Log	Current DCL verify value

Description

Use this command to enable or disable LogMiner operations on an Oracle Rdb database. When LogMiner is enabled, the Oracle Rdb database software writes additional information to the after-image journal file when records are added, modified, and deleted from the database. This information is used during the unload operation.

Command Parameters

root-file-spec
The root file specification of the database. The default file extension is .rdb.

Command Qualifiers

Disable
Specifies that LogMiner operations are to be disabled for the database. When LogMiner is disabled, the Oracle Rdb software does not journal information required for LogMiner operations. When LogMiner is disabled for a database, the RMU Unload After_Journal command is not functional on that database.

Enable
Specifies that LogMiner operations are to be enabled for the database. When LogMiner is enabled, the Oracle Rdb database software logs additional information to the after-image journal file. This information allows LogMiner to extract records. The database must already have after-image journaling enabled.

Log
Nolog
Specifies that the setting of the LogMiner state for the database be reported to SYS$OUTPUT. The default is the setting of the DCL VERIFY flag, which is controlled by the DCL SET VERIFY command.

Usage Notes

- To use the RMU Set Logminer command, you must have the RMU$BACKUP, RMU$RESTORE, or RMU$ALTER privilege in the root file access control list (ACL) for the database or the OpenVMS SYSPRV or BYPASS privilege.
- The RMU Set Logminer command requires offline access to the database. The database must be closed and no other users may be accessing the database.
- Execute a full database backup operation after issuing an RMU Set Logminer command that displays the RMU-W-DOFULLBCK warning message. Immediately after enabling LogMiner, you should perform a database after-image journal backup using the RMU Backup After_Journal command.

Examples

Example 1

The following example enables a database for LogMiner for Rdb operation.

```
$ RMU /SET LOGMINER /ENABLE OLTPDB.RDB
```

7.2 RMU Unload After_Journal Command

Allows you to extract added, modified, and deleted record contents from committed transactions from specified tables in one or more after-image journal files.

Format

```
RMU/Unload/After_Journal root-file-spec aij-file-name
```

Command Qualifiers	Defaults
/Before=date-time	None
/Extend_Size=integer	/Extend_Size=1000
/IO_Buffers=integer	/IO_Buffers=2
/[No]Log	Current DCL verify value
/Options=File=file-spec	See description
/Output=file-spec	/Output=SYS$OUTPUT
/Select=selection-type	/Select=Commit_Transaction
/Since=date-time	None
/Sort_Workfiles=integer	/Sort_Workfiles=2
/Statistics_Interval=integer	See description
/Table=(Name=table-name, [table-options ...])	See description
/[No]Trace	/Notrace

Description

The RMU Unload After_Journal command translates the binary data record contents of an after-image journal (.aij) file into an output file. Data records for the specified tables for committed transactions are extracted to an output stream (file, device, or application callback) in the order that the transactions were committed.

To use the RMU Unload After_Journal command, you must have first enabled the database for LogMiner extraction. Use the RMU Set Logminer command to enable the LogMiner for Rdb feature for the database. See the Section 7.1 for more information.

Data records extracted from the .aij file are those records that transactions added, modified, or deleted in base database tables. Index nodes, database metadata, segmented strings (BLOB), views, COMPUTED BY columns, system records, and temporary tables cannot be unloaded from after-image journal files.

Only the final content of a record for each transaction is extracted. Multiple changes to a single record within a transaction are condensed so that only the last revision of the record appears in the output stream. It is not possible to determine which columns were changed in a data record directly from the after-image journal file. The record itself would have to be compared to the content of a previous record in order to determine which columns were changed.

The database used to create the after-image journal files being extracted must be available during the RMU Unload After_Journal command execution. The database is used to obtain metadata information (such as table names, column counts, record version, and record compression) needed to extract data records from the .aij file. The database may be accessed either locally (on the same computer system) or remotely (over a network connection). The database is used only as a metadata reference. The database is read solely to load the metadata and is then detached.

The after-image journal file or files are processed sequentially, and all specified tables are extracted in one pass through the after-image journal file.

As each transaction commit record is processed, all modified and deleted records for the specified tables are sorted to remove duplicates and then the modified and deleted records are written to the output streams. Transactions that were rolled back are discarded. Data records for tables not being extracted are discarded. The actual order of output records within a transaction is not predictable.

In the extracted output, records that were modified or added are returned as being modified. It is not possible to distinguish between inserted and updated records in the output stream. Deleted (erased) records are returned as being deleted. A transaction that modifies and deletes a record generates only a deleted record. A transaction that adds a new record to the database and then deletes it within the same transaction generates only a deleted record.

LogMiner signals that a row has been deleted by placing a D in the RDB$LM_ACTION field and then recording the contents of the row at the instant before the delete operation in the user fields of the output record. If a row was modified several times within a transaction before being deleted, the output record will contain only the delete indicator and the results of the last modify operation. If a row is inserted and deleted in the same transaction, only the delete record appears in the output.

Records from multiple tables may be output to the same or to different destination streams. Possible output destination streams include the following:

- File
- OpenVMS Mailbox
- OpenVMS Pipe
- Direct callback to an application through a run-time activated sharable image

Command Parameters

root-file-spec
The root file specification of the database for the after-image journal file from which tables will be unloaded. The default file extension is .rdb.

The database must be the same database that was used to create the after-image journal files. The database is required so that the table metadata (information about data) is available to the RMU Unload After_Journal command. In particular, the names and relation identification of valid tables within the database is required along with the number of columns in the table and the compression information for the table in various storage areas.

The process attaches to the database briefly at the beginning of the extraction operation in order to read the metadata. Once the metadata has been read, the process disconnects from the database for the remainder of the operation.

aij-file-name
One or more input after-image journal backup files to be used as the source of the extraction operation. Multiple journal files can be extracted by specifying a comma-separated list of file specifications. OpenVMS wildcard specifications (using the * and % characters) are supported to extract a group of files. A

file specification beginning with the at (@) character refers to an options file containing a list of after-image journal files (rather than the file specification of an after-image journal itself). If you use the at (@) character syntax, you must enclose the at (@) character and the file name in double quotation marks (for example, specify aij-file-name as "@files.opt"). The default file extension is .aij.

Command Qualifiers

Before=date-time

Specifies the ending time and date for transactions to be extracted. Based on the Select qualifier, transactions that committed or started prior to the specified Before date are selected. Information changed due to transactions that committed or started after the Before date is not included in the output.

Extend_Size=integer

Specifies the file allocation and extension quantity for the output data files. The default extension size is 1000 blocks. Using a larger value can help reduce output file fragmentation and can improve performance when large amounts of data are extracted.

IO_Buffers=integer

Specifies the number of I/O buffers used for the output data files. The default number of buffers is 2. The default value is generally adequate. With sufficiently fast I/O subsystem hardware, additional buffers may improve performance. However, using a larger number of buffers will also consume additional virtual memory and process working set.

Log
Nolog

Specifies that the extraction of the .aij file be reported to SYS$OUTPUT or the destination specified with the Output qualifier. When activity is logged, the output from the Log qualifier provides the number of transactions committed and rolled back. The default is the setting of the DCL VERIFY flag, which is controlled by the DCL SET VERIFY command.

Options=File=file-spec

An options file contains a list of tables and output destinations. The options file may be used instead of, or along with, the Table qualifier to specify the tables to be extracted. Each line of the options file must specify a table name prefixed with "Table=". After the table name, the output destination is specified as either "Output=" or "Callback_Module=" and "Callback_Routine=".

```
TABLE=tblname,CALLBACK_MODULE=image,CALLBACK_ROUTINE=routine
TABLE=tblname,OUTPUT=outfile
```

The Record_Definition=file-spec option from the Table qualifier can be used to create a record definition file for the output data. The default file type is .rrd and the default file name is the name of the table.

The Table_Definition=file-spec option from the Table qualifier can be used to create a file with an SQL statement to create a table to hold transaction data. The default file type is .sql and the default file name is the name of the table.

Each option in the Options=File qualifier must be fully specified (no abbreviations are allowed) and followed with an equal sign (=) and a value string. The value string must be followed by a comma or the end of a line. Continuation lines may be specified by using a trailing dash. Comments are indicated by using the exclamation point (!) character.

Output=file-spec
Redirects the log and trace output (selected with the Log and Trace qualifiers) to the named file. If this qualifier is not specified, the output generated by the Log and Trace qualifiers, which can be voluminous, is displayed to SYS$OUTPUT.

Select=selection-type
Specifies if the date and time of the Before and Since qualifiers refers to transaction start time or transaction commit time.

The following options can be specified as the selection-type of the Select qualifier:

- Commit_Transaction

  Specifies that the Before and Since qualifiers select transactions based on the time of the transaction commit.

- Start_Transaction

  Specifies that the Before and Since qualifiers select transactions based on the time of the transaction start.

The default for date selection is Commit_Transaction.

Since=date-time
Specifies the starting time for transactions to be extracted. Based on the Select qualifier, transactions that committed on or after the specified Since date are selected. Information from transactions that committed or started prior to the specified Since date is not included in the output.

Sort_Workfiles=integer
Specifies the number of sort work files. The default number of sort work files is 2. When large transactions are being extracted, using additional sort work files may improve performance by distributing I/O loads over multiple disk devices. Use the SORTWORKn (where n is a number from 0 to 9) logical names to specify the location of the sort work files.

Statistics_Interval=integer
Specifies that statistics are to be displayed at regular intervals so that you can evaluate the progress of the unload operation.

The displayed statistics include:

- Elapsed time
- CPU time
- Buffered I/O
- Direct I/O
- Page faults
- Number of records unloaded for a table

If the Statistics_Interval qualifier is specified, the default interval is 60 seconds (1 minute). The minimum value is 1 second. If the unload operation completes successfully before the first time interval has passed, you will receive an informational message on the number of files unloaded. If the unload operation is unsuccessful before the first time interval has passed, you will receive error messages and statistics on the number of records unloaded.

At any time during the unload operation, you can press Ctrl/T to display the current statistics.

Table=(Name=table-name, table-options)
Specifies the name of a table to be unloaded and an output destination. The table-name must be a table within the database. Views, indexes, and system tables may not be unloaded from the after-image journal file.

The following table-options can be specified with the Table qualifier:

- Output=file-spec

  If an Output file specification is present, unloaded records are written to the specified location.

- Callback_Module=image-name, Callback_Routine=routine-name

  LogMiner for Rdb uses the OpenVMS library routine LIB$FIND_IMAGE_SYMBOL to activate the specified sharable image and locate the specified entry point routine name. This routine will be called with each extracted record. A final call is made with the "Action" field set to "E" to indicate the end of the output stream. These options must be specified together.

- Record_Definition=file-spec

  The Record_Definition=file-spec option can be used to create a record definition .rrd file for the output data. The default file type is .rrd and the default file name is the name of the table.

- Table_Definition=file-spec

  The Table_Definition=file-spec option can be used to create a file with an SQL statement to create a table to hold transaction data. The default file type is .sql and the default file name is the name of the table.

Note that, unlike other qualifiers where only the final occurrence of the qualifier is used by an application, the Table qualifier may be specified multiple times for the RMU Unload After_Journal command. Each occurrence of the Table qualifier must specify a different table.

Trace
NoTrace
Specifies that the unloading of the .aij file be traced. The default is Notrace. When the unload operation is traced, the output from the Trace qualifier identifies transactions in the .aij file by transaction sequence numbers (TSNs) and describes what Oracle RMU did with each transaction during the unload process. You can specify the Log qualifier with the Trace qualifier.

Usage Notes

- To use the RMU Unload After_Journal command for a database, you must have the RMU$DUMP privilege in the root file access control list (ACL) for the database or the OpenVMS SYSPRV or BYPASS privilege.
- You can only extract changed records from a backup copy of the after-image journal files. You create this file using the RMU Backup After_Journal command. You also cannot extract from an .aij file that has been optimized with the RMU Optimize After_Journal command. And, you cannot extract an active, primary .aij file.

- As part of the extraction process, Oracle RMU sorts extracted journal records to remove duplicate record updates. Because .aij file extraction uses the OpenVMS Sort/Merge Utility (SORT/MERGE) to sort journal records, you can improve the efficiency of the sort operation by changing the number and location of the work files used by SORT/MERGE. The number of work files is controlled by the Sort_Workfiles qualifier of the RMU Unload After_Journal command. The allowed values are 1 through 10 inclusive, with a default value of 2. The location of these work files can be specified with device specifications, using the SORTWORKn logical name (where n is a number from 0 to 9). See the OpenVMS documentation set for more information on using SORT/MERGE. See the *Oracle Rdb7 Guide to Database Performance and Tuning* for more information on using these Oracle Rdb logical names.
- You can redirect the .aij rollforward temporary work files to a different disk and directory location than the current default directory by assigning a different directory to the RDM$BIND_AIJ_WORK_FILE logical name in the LNM$FILE_DEV name table. This can help to alleviate I/O bottlenecks that might occur on the default disk.
- The RMU Unload After_Journal command can read either a backed up .aij file on disk or a backed up .aij file on tape that is in the Old_File format.
- One or more tables can be selected to be extracted from an after-image journal file. All tables specified by the Table qualifier and all those specified in the Options file are combined to produce a single list of output streams. A particular table may be specified only once. Multiple tables may be written to the same output destination by specifying the exact same output stream specification (that is, by using an identical file specification).
- At the completion of the unload operation, RMU creates a number of DCL symbols that contain information about the extraction statistics. For each table extracted, RMU creates the following symbols:
  - RMU$UNLOAD_DELETE_COUNT_tablename
  - RMU$UNLOAD_MODIFY_COUNT_tablename
  - RMU$UNLOAD_OUTPUT_tablename

  The tablename component of the symbol is the name of the table. When multiple tables are extracted in one operation, multiple sets of symbols are created. The value for the symbols RMU$UNLOAD_MODIFY_COUNT_tablename and RMU$UNLOAD_DELETE_COUNT_tablename is a character string containing the number of records returned for modified and deleted rows. The RMU$UNLOAD_OUTPUT_tablename symbol is a character string indicating the full file specification for the output destination, or the sharable image name and routine name when the output destination is an application callback routine.
- When using the Callback_Module and Callback_Routine option, you must supply a sharable image with a universal symbol or entry point for LogMiner to be able to call your routine. See the OpenVMS manual discussing the Linker utility for more information about creating sharable images.
- Your Callback_Routine will be called once for each output record. The Callback_Routine will be passed two parameters:
  - The length of the output record, by longword value
  - A pointer to the record buffer

The record buffer is a data structure of the same fields and lengths written to an output destination.

- Because the Oracle RMU image is a known image, your sharable image must also be a known image. Use the OpenVMS Install Utility to make your sharable image known. You may wish to establish an exit handler to perform any required cleanup processing at the end of the extraction.

Examples

Example 1

The following example unloads the EMPLOYEES table from the .aij backup file MFP.AIJBCK.

```
RMU /UNLOAD /AFTER_JOURNAL MFP.RDB MFP.AIJBCK -
    /TABLE = (NAME = EMPLOYEES, OUTPUT = EMPLOYEES.DAT)
```

Example 2

The following example simultaneously unloads the SALES, STOCK, SHIPPING, and ORDERS tables from the .aij backup files MFS.AIJBCK_1-JUL-1999 through MFS.AIJBCK_3-JUL-1999. Note that the input .aij backup files are processed sequentially in the order specified.

```
$ RMU /UNLOAD /AFTER_JOURNAL MFS.RDB -
   MFS.AIJBCK_1-JUL-1999, -
   MFS.AIJBCK_2-JUL-1999, -
   MFS.AIJBCK_3-JUL-1999 -
   /TABLE = (NAME = SALES, OUTPUT = SALES.DAT) -
   /TABLE = (NAME = STOCK, OUTPUT = STOCK.DAT) -
   /TABLE = (NAME = SHIPPING, OUTPUT = SHIPPING.DAT) -
   /TABLE = (NAME = ORDER, OUTPUT = ORDER.DAT)
```

Example 3

To unload data based on a time range, use the Before and Since qualifiers. The following example extracts changes made to the PLANETS table by transactions that committed between 1-SEP-1999 at 14:30 and 3-SEP-1999 at 16:00.

```
$ RMU /UNLOAD /AFTER_JOURNAL MFS.RDB MFS.AIJBCK -
   /TABLE = (NAME = PLANETS, OUTPUT = PLANETS.DAT) -
   /BEFORE = "3-SEP-1999 16:00:00.00" -
   /SINCE = "1-SEP-1999 14:30:00.00"
```

Example 4

The following example simultaneously unloads the SALES and STOCK tables from all .aij backup files that match the wildcard specification MFS.AIJBCK_1999-07-*. The input .aij backup files are processed sequentially in the order returned from the file system.

```
$ RMU /UNLOAD /AFTER_JOURNAL MFS.RDB -
   MFS.AIJBCK_1999-07-* -
   /TABLE = (NAME = SALES, OUTPUT = SALES.DAT) -
   /TABLE = (NAME = STOCK, OUTPUT = STOCK.DAT)
```

Example 5

The following example unloads the TICKER table from the .aij backup files listed in the file called AIJ_BACKUP_FILES.DAT (note the double quotation marks surrounding the at (@) character and the file specification). The input .aij backup files are processed sequentially. The output records are written to the mailbox device called MBA127:. A separate program is already running on the system, and it reads and processes the data written to the mailbox.

```
$ RMU /UNLOAD /AFTER_JOURNAL MFS.RDB -
   "@AIJ_BACKUP_FILES.DAT" -
   /TABLE = (NAME = TICKER, OUTPUT = MBA127:)
```

Example 6

To move transaction data from one database into a change table in another database, you can use the RMU Unload After_Journal command followed by RMU Load commands. A record definition (.rrd) file would need to be created for each table being loaded into the target database. The record definition files can be created by specifying the Record_Definition option on the Table qualifier.

```
$ RMU /UNLOAD /AFTER_JOURNAL OLTP.RDB MYAIJ.AIJBCK -
  /TABLE = ( NAME = MYTBL, -
             OUTPUT = MYTBL.DAT, -
             RECORD_DEFINITION=MYLOGTBL) -
  /TABLE = ( NAME = SALE, -
             OUTPUT=SALE.DAT, -
             RECORD_DEFINITION=SALELOGTBL)

$ RMU /LOAD WAREHOUSE.RDB MYLOGTBL MYTBL.DAT -
   /RECORD_DEFINITION = FILE = MYLOGTBL.RRD

$ RMU /LOAD WAREHOUSE.RDB SALELOGTBL SALE.DAT -
   /RECORD_DEFINITION = FILE = SALELOGTBL.RRD
```

Example 7

Instead of the Table qualifier, an Options file can be used to specify the table or tables to be extracted, as shown in the following example.

```
$ TYPE TABLES.OPTIONS
TABLE=MYTBL, OUTPUT=MYTBL.DAT
TABLE=SALES, OUTPUT=SALES.DAT
$ RMU /UNLOAD /AFTER_JOURNAL OLTP.RDB MYAIJ.AIJBCK -
   /OPTIONS = FILE = TABLES.OPTIONS
```

7.3 Restrictions and Limitations with LogMiner for Rdb

The following restrictions exist for the LogMiner for Rdb feature:

- Temporary tables cannot be extracted. Modifications to temporary tables are not written to the after-image journal file and, therefore, are not available to LogMiner for Rdb.
- Optimized after-image journal files cannot be used as input to the LogMiner for Rdb. Information needed by the RMU Unload After_Journal command is removed by the optimization process.
- Records removed from tables using the SQL TRUNCATE TABLE statement are not extracted. The SQL TRUNCATE TABLE statement does not journal each individual data record being removed from the database.
- Records removed by dropping tables using the SQL DROP TABLE statement are not extracted. The SQL DROP TABLE statement does not journal each individual data record being removed from the database.
- Tables that use the vertical record partitioning (VRP) feature cannot be extracted using LogMiner for Rdb. LogMiner software currently does not detect these tables. A future release of Oracle Rdb will detect and reject access to vertically partitioned tables.
- Segmented string data (BLOB) cannot be extracted using LogMiner for Rdb. Because the segmented string data is related to the base table row by means of a database key, there is no convenient way to determine what data to extract. Additionally, the data type of an extracted column is changed from LIST OF BYTE VARYING to BIGINT. This column contains the DBKEY of the original BLOB data. Therefore, the contents of this column should be considered unreliable.
- COMPUTED BY columns in a table are not extracted. These columns are not stored in the after-image journal file.
- VARCHAR fields are not space padded in the output file. The VARCHAR data type is extracted as a 2-byte count field and a fixed-length data field. The 2-byte count field indicates the number of valid characters in the fixed-length data field. Any additional contents in the data field are unpredictable.
- You cannot extract changes to a table when the table definition is changed within an after-image journal file. Data definition language (DDL) changes to a table are not allowed within an .aij file being extracted. All records in an .aij file must be the current record version. If you are going to perform DDL operations on tables that you wish to extract using the LogMiner for Rdb, you should:
  1. Back up your after-image journal files.
  2. Extract the .aij files using the RMU Unload After_Journal command.
  3. Make the DDL changes.
- Do not use the OpenVMS Alpha High Performance Sort/Merge utility (selected by defining the logical name SORTSHR to SYS$SHARE:HYPERSORT) when using LogMiner for Rdb. HYPERSORT supports only a subset of the library sort routines that LogMiner requires. Make sure that the SORTSHR logical name is not defined to HYPERSORT.

7.4 Information Returned by LogMiner for Rdb

LogMiner for Rdb appends several output fields to the data fields, creating an output record. The output record contains fixed-length fields in a binary data format (that is, integer fields are not converted to text strings). The data fields correspond to the extracted table columns. This information may or may not be required by all applications and readers of the data. There is currently no available method to restrict or reorder the output fields.

Extracted data field contents are the fields that are actually stored in the Oracle Rdb database. COMPUTED BY fields are not extracted because they are not stored in the database or in the after-image journal file. Segmented string (BLOB) contents are not extracted.

Table 7–1 describes the output fields and data types of an output record.

Table 7–1 Output Fields

Field Name	Data Type	Description
ACTION	CHAR (1 byte)	Indicates record state. "M" indicates an insert or modify action. "D" indicates a delete action. "E" indicates stream end-of-file (EOF) when a callback routine is being used.
RELATION_NAME	CHAR (31 bytes)	Table name. Space padded to 31 characters.
RECORD_TYPE	LONGWORD INTEGER	The Oracle Rdb internal relation identifier.
DATA_LEN	WORD INTEGER	Length, in bytes, of the data record content.
NBV_LEN	WORD INTEGER	Length, in bits, of the null bit vector content.
DBK	DBKEY (64-bit QUADWORD)	Records logical database key. The database key is a 3-field structure containing a 16-bit line number, a 32-bit page number and a 16-bit area number.
START_TAD	DATE VMS	Date/time of the start of the transaction.
COMMIT_TAD	DATE VMS	Date/time of the commitment of the transaction.
TSN	QUADWORD INTEGER	Transaction sequence number of the transaction that performed the record operation.
RECORD_VERSION	WORD INTEGER	Record version.
Record Data	Varies	Actual data record field contents.

(continued on next page)

Table 7–1 (Cont.) Output Fields

Field Name	Data Type	Description
Record NBV	BIT VECTOR (array of bits)	Null bit vector. There is one bit for each field in the data record. If a bit value is 1, the corresponding field is NULL; if a bit value is 0, the corresponding field is not NULL and contains an actual data value. The null bit vector begins on a byte boundary. Any extra bits in the final byte of the vector after the final null bit are unused.

7.5 Record Definition Prefix for LogMiner Fields

An RMS file containing the record structure definition for the output file can be used as an input file to the RMU Load command if extracted data is to be loaded into an Oracle Rdb database. The record description uses the CDO record and field definition format (this is the format used by the RMU Load and RMU Unload commands when the Record_Definition qualifier is used). The default file extension is .rrd.

The record definition for the fields that LogMiner for Rdb writes to the output is shown in the following example. These fields can be manually appended to a record definition file for the actual user data fields being unloaded. Alternately, the Record_Definition qualifier can be used with the Table qualifier or within an Options file to automatically create the record definition file. This can be used to load a transaction table within a database. A **transaction table** is the output that LogMiner for Rdb writes to a table consisting of sequential transactions performed in a database.

```
DEFINE FIELD RDB$LM_ACTION          DATATYPE IS TEXT SIZE IS 1.
DEFINE FIELD RDB$LM_RELATION_NAME      DATATYPE IS TEXT SIZE IS 31.
DEFINE FIELD RDB$LM_RECORD_TYPE     DATATYPE IS SIGNED LONGWORD.
DEFINE FIELD RDB$LM_DATA_LEN        DATATYPE IS SIGNED WORD.
DEFINE FIELD RDB$LM_NBV_LEN         DATATYPE IS SIGNED WORD.
DEFINE FIELD RDB$LM_DBK             DATATYPE IS SIGNED QUADWORD.
DEFINE FIELD RDB$LM_START_TAD       DATETYPE IS DATE
DEFINE FIELD RDB$LM_COMMIT_TAD      DATATYPE IS DATE
DEFINE FIELD RDB$LM_TSN             DATATYPE IS SIGNED QUADWORD.
DEFINE FIELD RDB$LM_RECORD_VERSION  DATATYPE IS SIGNED WORD.
```

7.6 SQL Table Definition Prefix for LogMiner Fields

The SQL record definition for the fields that LogMiner for Rdb writes to the output is shown in the following example. These fields can be manually appended to the table creation command for the actual user data fields being unloaded. Alternately, the Table_Definition qualifier can be used with the Table qualifier or within an Options file to automatically create the SQL definition file. This can be used to create a transaction table of changed data.

```
SQL> create table MYLOGTABLE (
cont> RDB$LM_ACTION          CHAR,
cont> RDB$LM_RELATION_NAME      CHAR (31),
cont> RDB$LM_RECORD_TYPE     INTEGER,
cont> RDB$LM_DATA_LEN        SMALLINT,
cont> RDB$LM_NBV_LEN         SMALLINT,
cont> RDB$LM_DBK             BIGINT,
cont> RDB$LM_START_TAD       DATE VMS,
cont> RDB$LM_COMMIT_TAD      DATE VMS,
cont> RDB$LM_TSN             BIGINT,
cont> RDB$LM_RECORD_VERSION  SMALLINT ...);
```

7.7 Segmented String Columns

Segmented string (also called BLOB or LIST OF BYTE VARYING) column data is not extracted. However, the field definition itself is extracted as a quadword integer representing the database key of the original segmented string data. In generated table definition or record definition files, a comment is added indicating that the segmented string data type is not supported by LogMiner for Rdb.

7.8 Additional Examples

The following sections contain additional examples.

7.8.1 Example .rrd for the EMPLOYEES Table

The following example is the transaction table record definition (.rrd) file for the EMPLOYEES table from the PERSONNEL database:

```
DEFINE FIELD RDB$LM_ACTION          DATATYPE IS TEXT SIZE IS 1.
DEFINE FIELD RDB$LM_RELATION_NAME   DATATYPE IS TEXT SIZE IS 31.
DEFINE FIELD RDB$LM_RECORD_TYPE     DATATYPE IS SIGNED LONGWORD.
DEFINE FIELD RDB$LM_DATA_LEN        DATATYPE IS SIGNED WORD.
DEFINE FIELD RDB$LM_NBV_LEN         DATATYPE IS SIGNED WORD.
DEFINE FIELD RDB$LM_DBK             DATATYPE IS SIGNED QUADWORD.
DEFINE FIELD RDB$LM_START_TAD       DATATYPE IS DATE.
DEFINE FIELD RDB$LM_COMMIT_TAD      DATATYPE IS DATE.
DEFINE FIELD RDB$LM_TSN             DATATYPE IS SIGNED QUADWORD.
DEFINE FIELD RDB$LM_RECORD_VERSION  DATATYPE IS SIGNED WORD.

DEFINE FIELD EMPLOYEE_ID            DATATYPE IS TEXT SIZE IS 5.
DEFINE FIELD LAST_NAME              DATATYPE IS TEXT SIZE IS 14.
DEFINE FIELD FIRST_NAME             DATATYPE IS TEXT SIZE IS 10.
DEFINE FIELD MIDDLE_INITIAL         DATATYPE IS TEXT SIZE IS 1.
DEFINE FIELD ADDRESS_DATA_1         DATATYPE IS TEXT SIZE IS 25.
DEFINE FIELD ADDRESS_DATA_2         DATATYPE IS TEXT SIZE IS 20.
DEFINE FIELD CITY                   DATATYPE IS TEXT SIZE IS 20.
DEFINE FIELD STATE                  DATATYPE IS TEXT SIZE IS 2.
DEFINE FIELD POSTAL_CODE            DATATYPE IS TEXT SIZE IS 5.
DEFINE FIELD SEX                    DATATYPE IS TEXT SIZE IS 1.
DEFINE FIELD BIRTHDAY               DATATYPE IS DATE.
DEFINE FIELD STATUS_CODE            DATATYPE IS TEXT SIZE IS 1.
DEFINE RECORD EMPLOYEES.
   RDB$LM_ACTION .
   RDB$LM_RELATION_NAME .
   RDB$LM_RECORD_TYPE .
   RDB$LM_DATA_LEN .
   RDB$LM_NBV_LEN .
   RDB$LM_DBK .
   RDB$LM_START_TAD .
   RDB$LM_COMMIT_TAD .
   RDB$LM_TSN .
   RDB$LM_RECORD_VERSION .
   EMPLOYEE_ID .
   LAST_NAME .
   FIRST_NAME .
   MIDDLE_INITIAL .
   ADDRESS_DATA_1 .
   ADDRESS_DATA_2 .
   CITY .
   STATE .
   POSTAL_CODE .
   SEX .
   BIRTHDAY .
   STATUS_CODE .
END EMPLOYEES RECORD.
```

7.8.2 Callback Module for the EMPLOYEES Table

The following C source code segment demonstrates the structure of a module that can be used as a callback module and routine to process employee transaction information from LogMiner for Rdb. The routine, Employees_Callback, would be called by LogMiner for Rdb for each extracted record. Note that the final time the callback routine is called, the RDB$LM_ACTION field will be set to "E" to indicate the end of the output stream.

```
#include <stdio>

typedef unsigned char date_type[8];
typedef unsigned char dbkey_type[8];
typedef unsigned char tsn_type[8];

typedef struct {
    unsigned char       rdb$lm_action;
    char                rdb$lm_relation_name[31];
    unsigned int        rdb$lm_record_type;
    unsigned short int  rdb$lm_data_len;
    unsigned short int  rdb$lm_nbv_len;
    dbkey_type          rdb$lm_dbk;
    date_type           rdb$lm_start_tad;
    date_type           rdb$lm_commit_tad;
    tsn_type            rdb$lm_tsn;
    unsigned short int  rdb$lm_record_version;
    char                employee_id[5];
    char                last_name[14];
    char                first_name[10];
    char                middle_initial[1];
    char                address_data_1[25];
    char                address_data_2[20];
    char                city[20];
    char                state[2];
    char                postal_code[5];
    char                sex[1];
    date_type           birthday;
    char                status_code[1];
} transaction_data;
```

```
void employees_callback (unsigned int data_len, transaction_data data_buf)
{    .
     .
     .
 return;}
```

Use the C compiler (either VAX C or DEC C) to compile this module. When linking this module, the symbol EMPLOYEES_CALLBACK needs to be externalized in the sharable image. Refer to the OpenVMS manual discussing the Linker utility for more information about creating sharable images.

On OpenVMS Alpha systems, you can use a LINK command similar to the following:

```
$ LINK /SHARABLE = EXAMPLE.EXE EXAMPLE.OBJ + SYS$INPUT: /OPTIONS
SYMBOL_VECTOR = (EMPLOYEES_CALLBACK = PROCEDURE)
<Ctrl/Z>
```

On OpenVMS VAX systems, you can use a LINK command similar to the following:

```
$ LINK /SHARABLE = EXAMPLE.EXE EXAMPLE.OBJ + SYS$INPUT: /OPTIONS
UNIVERSAL = EMPLOYEES_CALLBACK
<Ctrl/Z>
```

7.8.3 Using LogMiner and the RMU Load Command to Replicate Table Data

You can use triggers and a transaction table to construct a method to replicate table data from one database to another using RMU Unload After_Journal and RMU Load commands based on transactional changes to the source table. This data replication method requires no programming. Instead, existing features of Oracle Rdb can be combined to provide this functionality.

For this example, consider a simple customer information table called CUST with a unique customer ID value, customer name, address, and postal code. Changes to this table are to be moved from an OLTP database to a reporting database system on a periodic (perhaps nightly) basis.

First, in the reporting database, a customer table of the same structure as the OLTP customer table is created. In this example, this table is called RPT_CUST. It contains the same fields as the OLTP customer table called CUST.

```
SQL> CREATE TABLE RPT_CUST
  CUST_ID              INTEGER,
  CUST_NAME            CHAR (50),
  CUST_ADDRESS         CHAR (50),
  CUST_POSTAL_CODE     INTEGER);
```

Next, a temporary table is created in the reporting database for the LogMiner extracted transaction data from the CUST table. This temporary table definition specifies ON COMMIT DELETE ROWS so that data in the temporary table is deleted from memory at each transaction commit. A temporary table is used because there is no need to journal changes to the table.

```
SQL> CREATE GLOBAL TEMPORARY TABLE RDB_LM_RPT_CUST (
  RDB$LM_ACTION         CHAR,
  RDB$LM_RELATION_NAME  CHAR (31),
  RDB$LM_RECORD_TYPE    INTEGER,
  RDB$LM_DATA_LEN       SMALLINT,
  RDB$LM_NBV_LEN        SMALLINT,
  RDB$LM_DBK            BIGINT,
  RDB$LM_START_TAD      DATE VMS,
  RDB$LM_COMMIT_TAD     DATE VMS,
  RDB$LM_TSN            BIGINT,
  RDB$LM_RECORD_VERSION SMALLINT,
  CUST_ID               INTEGER,
  CUST_NAME             CHAR (50),
  CUST_ADDRESS          CHAR (50),
  CUST_POSTAL_CODE      INTEGER) ON COMMIT DELETE ROWS;
```

For data to be populated in the RPT_CUST table in the reporting database, a trigger is created for the RDB_LM_RPT_CUST transaction table. This trigger is used to insert, update, or delete rows in the RPT_CUST table based on the transaction information from the OLTP database for the CUST table. The unique CUST_ID field is used to determine if customer records are to be modified or added.

```
SQL> CREATE TRIGGER RDB_LM_RPT_CUST_TRIG
cont>  AFTER INSERT ON RDB_LM_RPT_CUST
cont>
cont> -- Modify an existing customer record
cont>
cont>  WHEN (RDB$LM_ACTION = 'M' AND
cont>        EXISTS (SELECT RPT_CUST.CUST_ID FROM RPT_CUST
cont>                WHERE RPT_CUST.CUST_ID = RDB_LM_RPT_CUST.CUST_ID))
cont>      (UPDATE RPT_CUST SET
cont>              RPT_CUST.CUST_NAME = RDB_LM_RPT_CUST.CUST_NAME,
cont>              RPT_CUST.CUST_ADDRESS = RDB_LM_RPT_CUST.CUST_ADDRESS,
cont>              RPT_CUST.CUST_POSTAL_CODE = RDB_LM_RPT_CUST.CUST_POSTAL_CODE
cont>       WHERE RPT_CUST.CUST_ID = RDB_LM_RPT_CUST.CUST_ID)
cont>  FOR EACH ROW
cont>
cont> -- Add a new customer record
cont>
cont>  WHEN (RDB$LM_ACTION = 'M' AND NOT
cont>        EXISTS (SELECT RPT_CUST.CUST_ID FROM RPT_CUST
cont>                WHERE RPT_CUST.CUST_ID = RDB_LM_RPT_CUST.CUST_ID))
cont>      (INSERT INTO RPT_CUST VALUES
cont>              (RDB_LM_RPT_CUST.CUST_ID,
cont>               RDB_LM_RPT_CUST.CUST_NAME,
cont>               RDB_LM_RPT_CUST.CUST_ADDRESS,
cont>               RDB_LM_RPT_CUST.CUST_POSTAL_CODE))
cont>  FOR EACH ROW
cont>
cont> -- Delete an existing customer record
cont>
cont>  WHEN (RDB$LM_ACTION = 'D')
cont>      (DELETE FROM RPT_CUST
cont>       WHERE RPT_CUST.CUST_ID = RDB_LM_RPT_CUST.CUST_ID)
cont>  FOR EACH ROW;
```

Within the trigger, the action to take (for example, to add, update, or delete a customer record) is based on the RDB$LM_ACTION field (which will be defined as D or M) and the existence of the customer record in the reporting database. For modifications, if the customer record does not exist, it is added; if it does exist, it is updated. For a deletion on the OLTP database, the customer record is deleted from the reporting database.

The RMU Load command is used to read the output from LogMiner for Rdb and load the data into the temporary table where each insert will result in the trigger executing. The Commit_Every qualifier is used to avoid filling memory with the customer records in the temporary table because as soon as the trigger executes, the record in the temporary table is no longer needed.

```
$ RMU /UNLOAD /AFTER_JOURNAL OLTP.RDB OLTP.AIJBCK -
      /TABLE = (NAME = CUST,
                OUTPUT = CUST.DAT,
                RECORD_DEFINITION = RDB_LM_RPT_CUST.RRD)
```

```
$ RMU /LOAD REPORT_DATABASE.RDB RDB_LM_RPT_CUST CUST.DAT -
      /RECORD_DEFINITION = FILE = RDB_LM_RPT_CUST.RRD -
      /COMMIT_EVERY = 1000
```

7.8.4 Using LogMiner to Minimize Application Downtime for Maintenance

Lengthy offline application or database maintenance operations can pose a significant problem in high-availability production environments. The LogMiner for Rdb feature can help reduce the length of downtime to a matter of minutes.

If a back up of the database is used for maintenance operations, the application can continue to be modified during lengthy maintenance operations. Once the maintenance is complete, the application can be shut down, the production system .aij file or files can be backed up, and LogMiner for Rdb can be used to extract changes made to production tables since the database was backed up. These changes can then be applied (using an application program or the trigger technique previously described) to the new database. Once the new database has been updated, the application can be restarted using the new database.

The sequence of events required would be similar to the following:

1. Perform a full online, quiet-point database backup of the production database.
2. Restore the backup to create a new database that will eventually become the production database.
3. Perform maintenance operations on the new database. (Note that the production system continues to run.)
4. Perform an online, quiet-point after-image journal backup of the production database.
5. Use the RMU Unload After_Journal command to unload all database tables into individual output files from the .aij backup file.
6. Using either the trigger technique or an application program, update the tables in the new database with the changed data.
7. Shut down the production application and close the database.
8. Perform an offline, quiet-point after-image journal backup of the production database.
9. Use the RMU Unload After_Journal command to unload all database tables into individual output files from the .aij backup file.
10. Using either the trigger technique or an application program, update the tables in the new database with the changed data.
11. Start an online, quiet-point backup of the new database.
12. Change logical names or the environment to specify the new database root file as the production database.

13. Restart the application on the new database.

Depending on the amount of application database activity, steps 4, 5, and 6 can be repeated to limit the amount of data that needs to be applied (and the amount of downtime required) during the final after-image journal backup and apply stage in steps 8, 9, and 10.

7.8.5 Using an OpenVMS Pipe

You can use an OpenVMS pipe to pass data from the RMU Unload After_Journal command to another application (for example, RMU Load). Do not use any options (such as the Log or Verify qualifiers) that could cause LogMiner to send extra output to the SYS$OUTPUT device, as that information would be part of the input data source stream to the next pipeline segment.

You may find that the OpenVMS default size of the pipe is too small if the records being extracted (including LogMiner fields) are larger than 256 bytes. If the pipe is too small, increase the SYSGEN parameters MAXBUF and DEFMBXMXMSG, and then reboot the system.

The following example uses LogMiner for Rdb to direct output to an OpenVMS pipe device and uses RMU Load to read the pipe device as the input data record stream. Using the pipeline allows parallel processing and also avoids the need for an intermediate disk file. Note that you must have created the record definition (.rrd) file prior to executing the command.

```
$ PIPE (RMU /UNLOAD /AFTER_JOURNAL OLTP.RDB AIJ1.AIJ -
    /TABLE = (NAME = MYTBL, OUTPUT = SYS$OUTPUT:)) -
     |  (RMU /LOAD REPORTS.RDB MYLOGTBL SYS$PIPE: -
        /RECORD_DEFINITION = FILE = MYLOGTBL.RRD)
```

A Implementing Row Cache

A.1 Overview

A.1.1 Introduction

Oracle Rdb uses buffers to temporarily store database pages during read and update operations. When you create or modify a database, you can set up buffers for database pages in either of the following ways:

- Local Buffers

  Database users have their own set of private local database page buffers. Data of interest is read from disk into a local database page buffer. Local buffers are not shared among users. Sharing occurs only when a database page is written back to disk and another user retrieves that database page. The sharing is done at the physical page level and can be I/O intensive.

- Global Buffers

  Database users on the same system share a common set of global database page buffers that reside in global memory. Database pages that are read from disk by one user can be seen directly by another user. Little or no I/O is needed to share global buffers; however, sharing data is still done at the level of database page buffers. A database page buffer has a fixed size across all storage areas in the database. The amount of data in a database page buffer that is of interest to multiple users may be small compared to its overall size. Although this model may be more efficient than using local buffers, there are better ways to share data among users.

Oracle Rdb offers a feature called row caching to enhance the performance of memory buffers. Because row caching is a cache of rows, you can use it in conjunction with local or global database page buffers. Please consider, however, that when using both global buffers and row cache, you could have two copies of data consuming your global memory—one copy in the row cache and one in a global buffer. Note also that row caches are not designed to be an “in-memory database”. As its name implies, a **row cache** is a set of database rows that reside in memory between the users and the rest of the database rows on disk. Data rows, system records, as well as hashed and sorted index nodes, can be cached. Access to a row in a row cache is through its logical database key (dbkey).

All processes attached to a database share a pool of row occurrences that reside in shared memory row caches. No disk I/O is needed to share a row in a row cache. Only the rows of interest, not the physical pages, are kept in shared memory, thereby increasing the use of shared memory. In addition, you can create many row caches, each with its own row size. Row caches can be used to efficiently store rows of specific sizes from specified tables. The Oracle Rdb implementation of row caches gives you the option to specify portions of row caches to occupy process private virtual memory, shared global pagefile sections on OpenVMS systems, or shared physical main memory. Oracle Rdb row caching also allows you to use very large memory (VLM) on OpenVMS Alpha systems. Subsequent sections provide more detail on each of these options.

The row caching feature is designed to improve performance through reduced I/O operations by finding rows of interest in the row cache instead of accessing them on disk. The greater number of times the data is located in the row cache, the more useful the cache is and better overall performance results.

The next section describes how row caching works with basic Oracle Rdb database functions.

A.1.2 Database Functions Using Row Cache

The following list describes how common database operations use the row caching feature.

- Fetching Data

  When you request a row from a database, Oracle Rdb first checks to see if the requested row is located in a row cache. If the row is in a row cache, the row is retrieved from the cache. If the row is not in a cache, Oracle Rdb checks the page buffer pool. If the row is not in the page buffer pool, Oracle Rdb performs a disk I/O operation to retrieve the row. The requested row is then inserted into the row cache, if possible.

- Storing Data

  When a new row is stored in the database, Oracle Rdb may perform a disk I/O operation to find space for the new row and get a dbkey for the row. Once space has been reserved on a database page, Oracle Rdb checks for a row cache in which to put the new row. The new row is inserted into a row cache, if possible.

- Modifying Data

  If a modification to a row in a cache causes the row to grow (replaces a null value, for example), then the database page must be modified to reserve additional space for that row. If the database page does not have room for the modified row, resulting in fragmentation, then the row is deleted from the cache. If the modification keeps the row the same size or makes it smaller, then the modified row remains in the cache and no database page is accessed. This means that the unused space on the page is not reclaimed and hence is not immediately available for reuse. Compressed rows and indexes that are modified are more likely to require database access than uncompressed ones.

- Deleting Data

  If the row is in a row cache, Oracle Rdb sets the length of the row to zero to erase it. It is not erased from the database page on disk immediately. Therefore, the deleted space is not reusable immediately.

- When snapshots are enabled

  During a read-only transaction, Oracle Rdb first checks to see if the row is in a row cache. If the row is found and is visible to the transaction, the row is returned from the row cache and no disk I/O operation is necessary. If the row is not visible, Oracle Rdb must find the visible version of this row in the snapshot file. Information stored in the row cache, however, can shorten the search and thereby reduce I/O operations to the snapshot file.

During a read/write transaction that is performing an update, Oracle Rdb writes the before-image of the data to the snapshot file. Oracle Rdb writes the before-image information out to the snapshot file each time a row in the user's row cache **working set** is modified. If a row falls out of the working set list and is remodified later in the transaction, the before-image information is written back to the snapshot file when the row re-enters the working set.

Global and local buffers use the least-recently used (LRU) replacement strategy for database pages. Row caching uses a modified form of the LRU replacement strategy. Each database user can protect the last 10 rows they accessed. This group of rows is referred to as a **working set**. Rows that belong to a working set are considered to be **referenced** and are not eligible for row replacement.

During a read/write transaction that performs a delete operation, the processing is the same as described in the previous paragraphs.

A.1.3 Writing Modified Rows to Disk

With row caching, many data modifications are performed on the in-memory copy of the data. Therefore, Oracle Rdb must have a way to write these rows to storage on disk.

The following list describes the ways that modified rows can be written back to the database page on disk.

- If the page on which a modified row resides is in the user's buffer pool and is already locked by the user when the update to that row must be recorded in the row cache, then the update is made to the row in the cache and on the database page.

  In this case, the row cache entry is not considered to be marked or modified. This situation occurs when a transaction is committed or when a row is flushed from a row cache.

- During an undo operation, the before-image of each modified row is placed on the database page.

  An **undo** operation occurs as part of an aborted SQL statement, transaction rollback, or database recovery of a terminated user's process.

- During a **redo** operation, the after-image of each modified row is stored on the database page only if recovering from a node failure. If recovering from a process failure, no redo is done for in-memory row cache modifications because the row cache memory is still valid and intact. (Changes made to database pages are still redone.)

- During a row cache checkpoint operation, all modified rows (or all rows) from the row caches are written to disk storage.

  This is the most common method of writing updated rows back to disk storage.

- During a row cache sweep operation, a set of modified rows are written back to the database from the row cache. After the rows are written back to disk, the space they occupied is considered selectable for reuse.

  A row cache sweep operation is initiated when a user process tries to insert rows into a row cache and finds no free space available.

A.1.4 Row Cache Checkpointing and Sweeping

Checkpointing and sweeping operations are critical in performing the operations necessary to write modified, committed rows back to disk from a row cache. The row cache server (RCS) process performs these tasks. There is one RCS process per database. Any failure of the RCS process forces the shutdown of the entire database.

To monitor the status of rows in a row cache, Oracle Rdb maintains a modification flag for every row in a cache to indicate which rows have been modified. The modification flags are shown in the following table:

Modification Flag	Meaning
Marked	The row has been modified in the row cache only. If this modification remains only in the row cache at the time the transaction is committed, then this marked flag indicates this row in the row cache is not reflected in the database.
Hot	The marked row has been modified since the last checkpoint.
Cold	The marked row has not been modified since the last checkpoint.

The RCS process performs three types of operations:

- Synchronous operations where the requester is waiting for the operation to complete

  The following are operations of this type:

  - The RCS process checkpoint operation that is part of an AIJ fast-commit checkpoint

    For example, if the RMU Checkpoint command with the Wait qualifier is issued, then the requester will wait for the RCS process to complete its checkpoint.

  - A checkpoint to the database for all row caches before certain database utility operations can begin

- Row cache checkpoint operations

  Checkpointing is a repetitive, time-driven event that writes rows from all row caches back to disk storage. The RCS process writes data to a cache backing file (.rdc) or directly to the database for each cache, depending on how the row cache was defined. The time interval at which a checkpoint occurs is also programmable. When the last user detaches from the database, the RCS process performs a final checkpoint operation to the database (never to the cache backing files). See Section A.4.2.1 for more details.

- Row cache sweep operations

  Sweeping is done to make space available in a particular row cache. When a transaction requests space and none is available, the RCS process sweeps marked rows back from the particular row cache to the database. It also resets row cache reference counts if your database has experienced some user process failures. This creates free memory for subsequent transactions to insert rows into each cache. This may never be necessary if checkpointing is done at appropriate intervals. See Section A.4.2.3 for more details.

The RCS process selects work requests based on their priority; synchronous operations are checked first, then checkpoints, followed by sweep operations.

If a database is opened manually, the RCS process is started as part of the open operation. If a database is opened automatically, the RCS, by default, is started when a row cache is referenced for the first time.

When the last user disconnects from the database (with the database open setting set to automatic) or when the database is closed manually, the RCS process performs a final checkpoint to the database. When this operation completes, all marked rows have been written back to the database. The RCS process writes out its checkpoint information to indicate that backing files are no longer needed if there is a need to recover from a node failure. At this time, the cache backing files, if any, are deleted by default. If you want to preserve the backing files and have them be reused at database startup, define the logical RDM$BIND_RCS_KEEP_BACKING_FILES to “1”.

Details of the RCS actions can be seen by creating an RCS process log file. Before opening the database, define the RDM$BIND_RCS_LOG_FILE system logical name to indicate the device, directory, and file name of the RCS process log file you want to create. If no device and directory are specified, the RCS log file is created in the same directory as that which contains the database root file.

A.1.5 Node and Process Failure Recovery

The following sections describe how the row cache feature interacts with node and process failure recovery.

To understand how database recovery works with row caches, you should understand the interactions that occur when writing to row caches, writing to the recovery-unit journal (RUJ) files, and writing to the after-image journal (AIJ) files. This interaction is identical to the interactions that occur among database page buffers, RUJ journaling, and AIJ journaling. For more information, see the *Oracle Rdb Guide to Database Performance and Tuning*.

The AIJ fast commit feature is a prerequisite for enabling row caching. This means that updates to the database are not flushed back to the database pages at the time a transaction is committed. In the case of row caching, the modified rows reside in the in-memory row caches. However, all after-image (updated rows) must be flushed to the AIJ file when the transaction is committed. In the event of a failure, the committed, updated rows can be reapplied to the database from the AIJ file.

Recovery-unit journaling is critical in ensuring that rows can be returned to their previous state when either a SQL statement or transaction rolls back or aborts abnormally. A row’s before-image must be preserved BEFORE any modification is made to a row on a database page or in a row cache. Before-images are placed in an in-memory RUJ buffer. Only when that buffer becomes full or when a modified page or modified row cache entry is being put back must the RUJ information first be synchronously written to the RUJ file. For a database without row caches, this means the write IO to the RUJ file must be performed before a database page containing a modified row can be written to disk.

With row caches, Oracle Rdb is frequently modifying only memory, not database pages. The requirement for RUJ information being written BEFORE a modification is put back into the row cache still exists. Writing synchronous IOs to the RUJ before modifying in-memory row caches doesn’t make muct sense. Oracle Rdb minimizes this behavior in two ways:

- A modification to a row cache entry is first done in a local copy. Only when this local copy of the row must be flushed back to the row cache is the RUJ information written out.

- The RUJ buffer resides in a system-wide, shared memory global section that is visible to the DBR process. Therefore the before-image rows don't have to be written to the RUJ file unless an uncommitted modification to a database page (a store or a modify bigger operation) is forced to disk or when the RUJ buffer overflows.

The global section created for the RUJ buffers will be about 256 VAX pages or 16 Alpha pages for each allowed user of a database. One global section is created for each database that has row caching enabled. To disable this optimization for databases with row caching enabled, define the logical name RDM$BIND_RUJ_GLOBAL_SECTION_ENABLED to "0" in the system logical name table.

You need to increase several OpenVMS system parameters, as follows:

- GBLSECTIONS

  Increase by the maximum number of Oracle Rdb databases open at one time on the system.

- GBLPAGES

  Increase by 256 times the maximum number of users for each database open at one time on the system.

- GBLPAGFIL

  Increase by 256 (on OpenVMS VAX systems) or by 16 (on OpenVMS Alpha systems), times the maximum number of users for each database open at one time on the system.

There is no additional virtual memory consumption for database users when the RUJ global buffers optimization is enabled; each user process continues to use the same amount of virtual memory (256 blocks) as when the optimization is not enabled.

Databases that do not have row caching enabled will not have optimization enabled for the RUJ buffer in a global section.

A.1.5.1 Process Failure

When a process terminates abnormally, Oracle Rdb activates a database recovery (DBR) process to recover the work done by the terminated user. The DBR process first performs transaction REDO, reapplying committed transactions' modifications to the database pages that had only been written to the AIJ file back to the database. Because the row cache memory is still in tact, in-memory row cache changes do not have to be redone during REDO. The DBR process then proceeds to UNDO the user's outstanding transaction. If the RUJ system-wide process buffers are enabled, the DBR process first writes the current RUJ buffer to the RUJ file. It then recovers the RUJ file by placing the before-image of each row back on the database page. If the dbkey for that row is also found in a row cache, the before-image is placed back into the row cache too.

A.1.5.2 Node Failure

There are several events that constitute node failure to Oracle Rdb:

- Machine or operating system fails
- The Oracle Rdb monitor process terminates unexpectedly
- The Oracle Rdb RCS process terminates unexpectedly
- An Oracle Rdb DBR process terminates unexpectedly
- The RMU Monitor Stop command is issued with the Abort=delprc qualifier

- The RMU Close command is issued with the Abort=delprc qualifier

All of these events cause all access to an Oracle Rdb database to cease immediately. Recovery from a node failure event is deferred until the next time the database is attached or opened. Even if the RMU Open command with the Row_Cache=disabled qualifier is executed next, this will initiate recovery from the node failure. It will not create nor populate the in-memory row caches during the recovery. Once recovery has finished, no row caches will be active while the database stays open in this manner.

Oracle Rdb has several schemes for recovering a database after a node failure. For a database without row caching enabled and without global buffers enabled, Oracle Rdb recovers from a node failure by creating one DBR process for each abnormally terminated user and these DBR processes recover the database in parallel. For a database without row caching enabled but with global buffers enabled, Oracle Rdb recovers one database user at a time by creating one DBR process at a time. For a database with row caching enabled, Oracle Rdb creates one DBR process and that process performs recovery for all the users.

For recovery from a node failure for a database with row caching enabled, the DBR process performs recovery in the following steps.

1. Recovers the backing files. For each row cache that is checkpointed to a backing file, the DBR process:
   - Reads each row from the backing file.
   - If the row has been updated (marked), then the DBR process writes this row back to the appropriate database page.
   - Inserts this row into the empty row cache in shared memory. If the database is opened with row caching disabled or if the system logical name RDM$BIND_DBR_UPDATE_RCACHE is defined to "0", then the row caches are not repopulated from the backing files.
   - Places this dbkey in a row cache dbkey list.
2. Performs a REDO operation from the oldest user checkpoint. This includes the RCS process checkpoint when the RCS process last checkpointed the row caches.
   - For each transaction rolled back, the DBR process discards the updates.
   - For each transaction committed, the DBR process reapplies those updates to the database pages.

     Please note that ALL committed transactions since the oldest checkpoint are applied, not just all committed transactions for the users who were active at the time of the node failure.
   - If DBR is re-populating the row caches and this dbkey is found in the row cache dbkey list, then this occurrence replaces the current one in the row cache. If a row in a mixed format area is erased, it is removed from the row cache and its dbkey is removed from the dbkey list. This is necessary to prevent the physical dbkey that may be reused for a different table or index from being placed in the prior occurrence's row cache.
   - Once the redo operation is completed, the DBR process updates all users' checkpoints to be the current AIJ end-of-file.

3. Performs the UNDO operation for each aborted user's incomplete transaction, if any. The DBR process reads the before-images from the user's RUJ file and writes them back to the database. If the dbkey also exists in a row cache, then the before-image is also written to its row cache entry.

A.1.5.3 The RCS Process and Database Recovery

Because the RCS process and the DBR process both access the row cache structures, they must coordinate their activities. When a DBR process is activated, it immediately notifies the RCS process of its existence using a lock. Then the RCS process aborts whatever request it is performing, requeues the request at the head of the appropriate queue, and waits for the database recovery activity to complete. Upon completion of database recovery, the RCS process resumes its operations by executing the next operation based on priority.

A.1.6 Considerations When Using the Row Cache Feature

This section contains further information on using the row cache feature.

- Hot Standby

  Row caching is not allowed to be active on the standby database. Because the AIJ file does not contain logical dbkeys, there is no way to maintain rows in the cache on the standby system. On the standby system, issue the RMU Open command with the Row_Cache=Disabled qualifier to open the database without activating row caching. If failover is necessary, simply close the standby database and reopen it normally. Your standby database will have row caches activated.

- Backing files

  If you are using row cache backing files, then do not use Hot Standby on the same machine as the master database. Both databases will attempt to use the same backing files.

  Similarly, do not attempt to use the same directory location for backing files for two or more databases if any of their row cache names are identical. Multiple databases will attempt to use the same backing files.

- Utilities that access the database pages directly

  Some RMU commands do not access data by logical dbkey but instead read the database pages directly. These commands cannot access the row caches directly. Oracle Rdb resolves this problem by having each command request the RCS process write all marked rows back to the database. The RMU operation waits for this task to complete.

  The RMU commands affected by this are:

  - Backup online
  - Analyze
  - Verify
  - Copy database online

  These operations may exhibit a delay in starting. If you specify the RMU log qualifier, Oracle Rdb will output a message when it is waiting for the RCS request and when the RCS request has completed. If your database's row caches are set to checkpoint to the database rather than to backing files, then this delay will be minimized.

- Sequential scans

When the execution strategy for a query is a sequential scan, Oracle Rdb scans the physical areas by performing the same I/O operations it would do if there were not any row caches. The major reasons for this are as follows:

- Oracle Rdb does not have a list of all dbkeys in an area; it materializes them by reading all pages and examining all lines on each page. However, data is returned from the row cache if it is found there. Although Oracle Rdb reads the database pages to find the dbkeys of rows in the table, it still needs to look in the cache to see if the row is there. A row in the cache contains more recent data than that which is on disk.
- There is no guarantee that all rows in a sequential scan can fit in a row cache. Row caches are often sized to include a percentage of the total number of rows where the most commonly used rows can be shared in memory.

  Oracle Rdb is designed to avoid populating the cache during a strict sequential scan. It is designed this way because otherwise a query performing a sequential scan of a table looking for just a few records would fill the cache with every record and might force existing data in the cache back to disk. This would result in a row cache filled with records that you do not need in the cache.

  However, note that a sequential *index* scan will populate the cache with data, index rows, or both.

- Snapshots enabled

  The Oracle Rdb snapshot mechanism of preserving a consistent view of the database for read-only transactions is not changed by the row cache feature. The before-images of rows needed by read-only transactions are preserved when read/write transactions write them to the snapshot files. Therefore, when snapshots are enabled, update operations are written to the rows in the row cache and the before-image of the row is written to disk. Oracle Rdb has optimized the snapshot mechanism with row caches, however, so that the performance of readers and writers may be better with row caches than without.

  The performance of row caches is typically much faster when snapshots are disabled. All of the disk I/O operations necessary to read and write to the snapshot file are eliminated. This is the ideal situation.

- Fragmented rows

  Fragmented rows are not stored in the row cache. They are created by fetching the fragments from the database and materializing them in process-private virtual memory.

- Vertical record partitioning

  When a logical cache is defined for a vertically partitioned table, each partition of a row is cached as a separate row cache entry. Only partitions that your query references and that can fit are inserted into the row cache.

- Unexpected storage area growth

  Oracle Rdb has optimized row caching to minimize the disk I/O operations required. Frequently operations are performed in-memory only. Having the faster performance of in-memory updates is beneficial. However, when you make modifications that keep a row at its current size or smaller, or you make deletions, the database page does not reflect the amount of space that is in use. Even though the row is logically smaller or erased from the database, it has not been physically removed from the database page. The space it occupies cannot be reused by another transaction until this row is finally written back to the database, usually by the RCS process during a sweep or checkpoint operation, depending on your row cache settings. Because of this, storage areas may grow larger than anticipated. If space reclamation is critical for some storage areas, then consider checkpointing their row caches to the database on a regular basis.

A.2 Requirements for Using Row Caches

To use the row cache feature, an Oracle Rdb database must meet the following configuration requirements:

- The number of cluster nodes must be one.
- After-image journaling must be enabled.
- Fast commit must be enabled.
- One or more row cache slots must be reserved.
- Row caching must be enabled.

Use the RMU Dump command with the Header qualifier to see if you have met the requirements for using row caches. In the following example, warnings are displayed for row cache requirements that have not been met.

```
$ RMU/DUMP/HEADER INVENTORY
   .
   .
   .
    Row Caches...
      - Active row cache count is 4
      - Reserved row cache count is 20
      - Checkpoint information
          Time interval is 10 seconds
          Default source is updated rows
          Default target is backing file
          Default backing file directory is "DISK1:[CACHE]"
      - WARNING: Maximum node count is 16 instead of 1
      - WARNING: After-image journaling is disabled
      - WARNING: Fast commit is disabled
   .
   .
   .
```

A.3 Designing and Creating a Row Cache

The following sections describe considerations for designing and creating row caches.

A.3.1 Reserving Slots for Row Caches

When you create a database, reserve enough row cache slots for both current and future needs. To reserve additional slots and to add or drop a row cache, the database must be closed.

Use the RESERVE n CACHE SLOTS clause of the CREATE DATABASE or ALTER DATABASE statement to reserve slots for row caches, as shown in the following example:

```
SQL> CREATE DATABASE FILENAME INVENTORY
   .
   .
   .
cont>  RESERVE 20 CACHE SLOTS;
```

If you do not specify a RESERVE n CACHE SLOTS clause, Oracle Rdb reserves one slot by default.

A.3.2 Row Cache Types

The two types of row caches are described in the following list:

- Physical area

  You can create a general row cache that is shared by all row types that reside in one or more storage areas. This is the basic type of row cache, called a **physical area row cache**. Because physical area row caches are defined for a storage area, multiple storage areas can map to the same physical area row cache. A physical area row cache can contain all row types in a storage area. In addition, when a physical area row cache is defined, all rows of different sizes in the specified storage area are candidates for the row cache.

  See Section A.3.2.1 for an example of how to assign a row cache to a storage area.

- Logical area

  You can create logical area row caches when you create a row cache by using the same name as an existing table or index. A **logical area row cache** is associated with all partitions, both horizontal and vertical, of a specific table or index.

  A logical area cache cannot store the system row from a database page in an mixed format area.

You can use both physical and logical caches to store a table and its index.

The following example shows the reason for using different caches for different row types. Assume the following sizes for the rows in a table and hashed index:

- System records of 16 bytes
- Hash buckets of 100 bytes
- Data rows of 320 bytes

If you created one cache for all three row types, with a row size of 320 bytes, much of the allocated memory would be wasted when storing the smaller system record and the hash bucket. Using this method, the amount of memory, excluding overhead, used for one row cache is as follows, assuming 15000 rows in the cache:

```
Total
number    = (# of rows in cache * row length of largest row)
of bytes

          = (15000 * 320)

          = 4800000 bytes
```

It is more efficient to have three caches, one for each of the row types:

- System records of 16 bytes (PARTS_SYS cache)
- Hash buckets of 100 bytes (PARTS_HASH cache)
- Data rows of 320 bytes (PARTS cache)

In this example the system records are stored in a physical cache (PARTS_SYS) while the hash index buckets and data rows are stored in logical caches (PARTS_HASH and PARTS).

The amount of memory, excluding overhead, used with three row caches is computed as follows:

```
Total
number    = (# of rows in cache * row length of system record) +
of bytes    (# of rows in cache * row length of hash bucket) +
            (# of rows in cache * row length of data row)

          = (5000 * 16) +
            (5000 * 100) +
            (5000 * 320)

          = 2180000 bytes
```

A.3.2.1 Assigning Storage Areas to Row Caches

When a storage area is associated with a row cache, the row cache can contain all types of rows, if they can fit. This is called a physical area row cache. One storage area can point to one row cache only. Multiple storage areas can be mapped to the same row cache.

You can also define a default row cache for all of the storage areas in the database by using one of the following statements:

- ALTER DATABASE ... ADD STORAGE AREA ... CACHE USING
- ALTER DATABASE .. ALTER STORAGE AREA ... CACHE USING
- CREATE DATABASE ... CREATE STORAGE AREA ... CACHE USING

The following example shows how to assign the same physical row cache to multiple storage areas:

```
SQL> ALTER STORAGE AREA
cont> PART_ID_A_E CACHE USING PARTS_SYS;
SQL> ALTER STORAGE AREA
cont> PART_ID_F_K CACHE USING PARTS_SYS;
```

A.3.2.2 Assigning Tables to Row Caches

A row cache is considered to be a logical area cache if its name is identical to the name of either a table or an index. If a logical area row cache is created for a vertically or horizontally partitioned table or horizontally partitioned index, then all rows in these partitions are mapped to the single logical area row cache. In the following example, a logical area cache called PARTS is created for the PARTS table that is horizontally partitioned across five storage areas:

```
SQL> CREATE STORAGE MAP PARTS_MAP FOR PARTS
cont> --
cont> -- Parts table partitioned by part_id
cont> --
cont>  STORE USING (PART_ID)
cont>      IN PART_ID_A_E WITH LIMIT OF ('F')
cont>      IN PART_ID_F_K WITH LIMIT OF ('L')
cont>      IN PART_ID_L_P WITH LIMIT OF ('Q')
cont>      IN PART_ID_Q_U WITH LIMIT OF ('V')
cont>      OTHERWISE IN PART_ID_V_Z
cont>      PLACEMENT VIA INDEX PARTS_HASH;
SQL>
   .
   .
   .
SQL> ALTER DATABASE FILENAME INVENTORY
cont>     ADD CACHE PARTS
cont>         ROW LENGTH IS 100 BYTES
cont>         CACHE SIZE IS 5000 ROWS;
```

Rows from all five partitions of the PARTS table are automatically cached in the PARTS row cache, if they can fit.

A.3.3 Sizing a Row Cache

When you size a row cache, you specify the following:

- Slot Size

  The slot size is the fixed length size of each entry in the row cache. This determines the size of the largest row that can be stored in the row cache. Oracle Rdb will not cache a row if it is larger than the cache's slot size. Use the ROW LENGTH IS parameter of the ADD, ALTER, or CREATE CACHE clause to specify the slot size of the row cache.

  Oracle Rdb automatically rounds up the row length to the next 4-byte boundary. This is done because longword aligned data structures perform optimally on its supported platforms.

  If you do not specify a slot size when creating a logical cache, Oracle Rdb generates a slot size based on the size of the table row or index node. Note, however, that Oracle Rdb finds the nominal row length of tables and indices using the area inventory page (AIP). Under certain circumstances this AIP length may not be the actual length of the row. In addition, some index structures may have no AIP entry at all. If no entry can be found, Oracle Rdb uses a default length of 256 bytes. Also, if the metadata for a table is modified, then the AIP length is not automatically updated. This can result in incorrect cache sizing. See the *Oracle Rdb Guide to Database Performance and Tuning* for more details on AIP lengths.

- Slot count

  The slot count is the number of rows that can be stored in the cache. Use the CACHE SIZE IS parameter of the ADD, ALTER, or CREATE CACHE clause to specify the number of rows that can be stored in the cache.

  If you do not specify the CACHE SIZE clause, Oracle Rdb creates a cache of 1000 rows by default.

The following example shows a row cache definition:

```
SQL> ADD CACHE PARTS
cont> ROW LENGTH IS 320 BYTES
cont> CACHE SIZE IS 3000 ROWS;
SQL> --
SQL> -- In this example, the slot size is 320 bytes
SQL> -- and the slot count is 3000.
SQL> --
```

It is important to select a proper slot size for the row cache. As stated previously, if a row is too large, Oracle Rdb will not cache the row. This can result in poor system performance because Oracle Rdb always checks the cache for the row before retrieving the row from disk. Use the RMU Dump Area command to determine the sizes of the data rows, hash buckets, and B-tree nodes. Keep in mind that row sizes within a table can vary greatly. If, for example, the largest row stored in a table is 100 bytes, but the majority of the rows range between 40 and 50 bytes, you may not necessarily want to choose 100 bytes for the slot size. However, you should account for most of the rows, including overhead. If you automatically select the largest row size without comparing it to the sizes of the other rows in the table, you might waste memory.

The following example dumps a few pages from the MY_AREA storage area:

```
$ RMU/DUMP/AREA=MY_AREA/START=5/END=10 TEST_DB/OUT=rmu_dump_area.out
```

Search the rmu_dump_area.out file for the occurrences of “total hash bucket” and “static data” as follows:

```
$ SEARCH RMU_DUMP_AREA.OUT "total hash bucket"
                                  ....  total hash bucket size: 97
                                  ....  total hash bucket size: 118
                                  ....  total hash bucket size: 118
                                  ....  total hash bucket size: 118
                                  ....  total hash bucket size: 118
                                  ....  total hash bucket size: 118
                                  ....  total hash bucket size: 118
                                  ....  total hash bucket size: 118
                                  ....  total hash bucket size: 118

   .
   .
   .
$ SEARCH rmu_dump_area.out "static data"
                                  ....  311 bytes of static data
                                  ....  311 bytes of static data
                                  ....  311 bytes of static data
                                  ....  311 bytes of static data
                                  ....  311 bytes of static data
                                  ....  311 bytes of static data
                                  ....  311 bytes of static data
                                  ....  311 bytes of static data
                                  ....  311 bytes of static data
                                  ....  311 bytes of static data
                                  ....  311 bytes of static data
                                  ....  311 bytes of static data
   .
   .
   .
```

The hash bucket size is 118 bytes and the data row size is 311 bytes. Other rows in this table may require more or less space. It is important to scan a representative sample of random pages to determine the appropriate row size. Oracle Rdb rounds row sizes up to the next longword.

The RMU Show Statistics row caching screens provide performance information on inserting rows into a cache. One of the statistics, “row too big”, indicates that a row is too large to fit into the specified cache. This statistic is also set when a row in a row cache becomes invalid and must be retrieved from the database page. For example, when a row in the row cache grows to the point where it becomes fragmented, it must be removed from the row cache. This is done by “redirecting” this row out of the row cache to disk, by setting its “row too big” attribute. See Section A.5.1 for more information on the RMU Show Statistics screens related to row caching.

The slot count multiplied by the slot size specifies the approximate size, in bytes, of the row cache. You should also take into account additional overhead. See Section A.3.4.1 for more information about sizing row caches.

A.3.4 Choosing Memory Location

When you create a row cache or modify a row cache definition, you have the option of specifying where in memory you want Oracle Rdb to create the cache. Row caches can reside in the following memory locations:

- Process global section on OpenVMS and shared memory partition on Digital UNIX.

  When you use global sections or shared memory created in the process space, you and other users share virtual memory and the operating system maps a cache to a private address space for each user.

  Use the SHARED MEMORY IS PROCESS parameter to specify that the cache be created in a process global section or shared memory partition as shown in the following example:

  ```
  SQL> ALTER DATABASE FILENAME MF_PERSONNEL
  cont> ADD CACHE EMPIDS_LOW_RCACHE
  cont> SHARED MEMORY IS PROCESS;
  ```

  This is the default.

- System space buffer

  The system space global section is located in the OpenVMS Alpha system space, which means that a system space global section is fully resident, or pinned in memory and does not affect the quotas of the working set of a process.

  System space is critical to the overall system. System space buffers are not paged; therefore, they use physical memory, thereby reducing the amount of physical memory available for other system tasks. This may be an issue if your system is constrained by memory. You should be careful when you allocate system space. Nonpaged dynamic pool (NPAGEDYN) and the VMScluster cache (VCC) are some examples of system parameters that use system space.

  Use the SHARED MEMORY IS SYSTEM parameter to specify that the cache be created in a system space buffer, as shown in the following example:

  ```
  SQL> ALTER DATABASE FILENAME MF_PERSONNEL
  cont> ADD CACHE EMPIDS_MID_RCACHE
  cont> SHARED MEMORY IS SYSTEM;
  ```

Consider allocating small caches that contain heavily accessed data in system space buffers. When a row cache is stored in a system space buffer, there is no process overhead and data access is very fast because the data does not need to be mapped to user windows. Also, OpenVMS Alpha Version 7 systems and later make additional system space available by moving page tables and balance slots into VLM space. The Hot Row Information screen in the RMU Show Statistics command displays a list of the most frequently accessed rows for a specific row cache.

- Very large memory

  Very large memory (VLM) on OpenVMS Alpha systems allows Oracle Rdb to use as much physical memory as is available on your system and to dynamically map it to the virtual address space of database users. VLM provides access to a large amount of physical memory through small virtual address windows. Even though VLM is defined in physical memory, the virtual address windows are defined and maintained in each user's private virtual address space or system space depending on the memory setting.

  Use the LARGE MEMORY parameter to specify that the cache be created in large memory.

  ```
  SQL> ALTER DATABASE FILENAME MF_PERSONNEL
  cont> ADD CACHE EMPIDS_OVER_RCACHE
  cont> LARGE MEMORY IS ENABLED;
  SQL>
  ```

  VLM is useful for large tables with high access rates. The only limiting factor with VLM is the amount of available physical memory on your system.

You view the physical memory through windows. You can specify the number of window panes with the WINDOW COUNT parameter. By default, Oracle Rdb allocates 100 window panes to a process.

Table A–1 summarizes the location in memory of each row cache object and whether process private virtual address windows are needed to access the data.

Table A–1 Memory Locations of Row Cache Objects

SHARED	LARGE	Control Structures	Data Rows	Windows
PROCESS[1]	DISABLED[3]	Process global section or shared memory partition	Process global section or shared memory partition	No
PROCESS[1]	ENABLED[4]	Process global section or shared memory partition	Physical memory	Yes
SYSTEM[2]	DISABLED[3]	System space	System space	No
SYSTEM[2]	ENABLED[4]	System space	Physical memory	Yes

[1]SHARED MEMORY IS PROCESS

- The row cache control structures are located in a process global section or shared memory partition.
- The storage of the data rows depends on whether large memory is enabled or disabled.
  - If large memory is enabled, data is stored in physical memory and windows from each user's process virtual address space are needed to access the data.
  - If large memory is disabled, data is stored in a process global section or shared memory partition and no windows are needed to access the data.

[2]SHARED MEMORY IS SYSTEM

- The row cache control structures are stored in system space.
- The storage of the data rows depends on whether large memory is enabled or disabled.
  - If large memory is enabled, data is stored in physical memory and windows from each user's process virtual address space are needed to access the data.
  - If large memory is disabled, data is stored in system space and no windows are needed to access the data.

[3]LARGE MEMORY IS DISABLED

- The storage of the data rows and the row cache control structures depends on whether shared memory is process or system.
  - If shared memory is process, the data and row cache control structures are stored in a process global section or shared memory partition and no windows are needed to access the data.
  - If shared memory is system, the data and row cache control structures are stored in system space and no windows are needed to access the data.

[4]LARGE MEMORY IS ENABLED

- The data rows are stored in physical memory and process private virtual address windows are needed to access the data.
- The storage of the row cache control structures depends on whether shared memory is process or system.
  - If shared memory is process, the control structures are stored in a process global section or shared memory partition.
  - If shared memory is system, the control structures are stored in system space.

It is important to consider the amount of memory available on your system before you start creating and using row caches.

On OpenVMS systems, you can use the DCL command SHOW MEMORY /PHYSICAL to check the availability and usage of physical memory. This command displays information on how much memory is used and how much is free. The free memory is available for VLM row caches in addition to user applications.

Because VLM row caches can consume a certain amount of system space for their virtual address windows, Oracle Corporation recommends that you define the VLM row caches first, so that any VLM system space requirements are satisfied before you define system space buffer row caches for small tables that contain frequently accessed data.

The following example shows a system that has 1.5 gigabytes of memory or a total of 196608 OpenVMS Alpha memory pages (an OpenVMS Alpha page is 8192 bytes):

```
$ SHOW MEMORY/PHYSICAL

              System Memory Resources on 29-MAY-1996 21:39:35.40

Physical Memory Usage (pages):     Total        Free      In Use    Modified
  Main Memory (1536.00Mb)         196608      183605       12657         346
```

Of the 1.5 gigabytes, 183605 pages remain on the free list. Most of this free memory is available for row cache allocation.

Assume a logical area cache has been defined for the MY_TABLE table. The following SQL statement maps the logical area cache:

```
SQL> ATTACH 'FILE TEST_DB';
SQL> SELECT * FROM MY_TABLE WHERE MY_HASH_INDEX = 100;
```

By issuing this SQL statement, the logical area cache has allocated the necessary memory accounting for 40462 OpenVMS Alpha pages, as shown in the following SHOW MEMORY/PHYSICAL command output:

```
$ SHOW MEMORY/PHYSICAL

              System Memory Resources on 29-MAY-1996 21:46:07.01

Physical Memory Usage (pages):     Total        Free      In Use    Modified
  Main Memory (1536.00Mb)         196608      143143       52766         699
```

Notice the amount of free memory has been reduced.

The following SHOW MEMORY/PHYSICAL command was issued after users attached to the database, allocated their working sets, and began to work:

```
              System Memory Resources on 29-MAY-1996 23:48:06.67
Physical Memory Usage (pages):     Total        Free      In Use    Modified
Main Memory (1536.00Mb)           196608       81046      112498        3064
```

In this example, only 81046 OpenVMS Alpha pages are left on the free list.

A.3.4.1 Sizing Considerations

The following information is intended to help you determine in which memory location to place your cache based on system resources. Generally, if your cache will fit into a process global section or system space buffer, then it will perform slightly better. If space is an issue, then you should place the cache in VLM.

When a cache is created in a process global section or system space buffer, Oracle Rdb sizes it using the following values:

- Each slot requires 48 bytes plus the length of the slot rounded to the next 4-byte boundary.
- Each cache requires a hash table of (4 * (the number of cache slots rounded to the next higher power of 2)) bytes.
- Each cache requires (24 * the maximum number of users) bytes.

When a cache is created in VLM, Oracle Rdb sizes it using the following values:

- Each slot requires 24 bytes plus the length of the slot rounded up to the next 4-byte boundary.

When VLM is enabled and the cache is created in a process global section or system buffer space, Oracle Rdb sizes it using the following values:

- Each slot requires 24 bytes.
- Each cache requires a hash table of (4 * (the number of cache slots rounded up to the next higher power of 2)) bytes.
- Each cache requires (24 * the maximum number of users) bytes.

The following example shows how Oracle Rdb sizes a cache containing 150,000 slots with a slot size of 500 bytes in a process global section or system space buffer and a maximum of 350 users. (Note that 2 to the 17th power is 262144.)

Example A–1 Sizing a Row Cache in a Global Section or System Space Buffer

```
Total
number  = (150000*(500+48)) + (262144*4) + (24*350)
of
bytes

        = 83,256,976 bytes
```

The following example shows how Oracle Rdb sizes the same cache in VLM.

Example A–2 Sizing a Row Cache in VLM

```
Total
number  = (150000*(500+24))
of
bytes

        = 78,600,000 bytes
```

The following example shows how Oracle Rdb sizes the same cache in a process global section or system space buffer with VLM enabled.

Example A–3 Sizing a Row Cache in Memory with VLM Enabled

```
Total
number  = (150000*24) + (262144*4) + (24*350)
of
bytes

        = 4,656,976 bytes
```

A.4 Using Row Cache

The following sections describe how to set parameters for the row cache feature.

A.4.1 Enabling and Disabling Row Cache

There are three ways in which Row Caching can be enabled and/or disabled.

1. You can enable row caching for a database by using the ROW CACHE IS ENABLED clause of the SQL ALTER DATABASE and CREATE DATABASE statements. The following example shows how to enable the row cache feature and its requirements:

   ```
   SQL> ALTER DATABASE FILENAME MF_PERSONNEL
   cont> NUMBER OF CLUSTER NODES IS 1
   cont> JOURNAL ENABLED (FAST COMMIT ENABLED)
   cont> RESERVE 20 CACHE SLOTS
   cont> ROW CACHE IS ENABLED;
   ```

   You can disable row caching for a database by using the ROW CACHE IS DISABLED clause of the SQL ALTER DATABASE and CREATE DATABASE statements:

   ```
   SQL> ALTER DATABASE FILENAME MF_PERSONNEL
   cont> ROW CACHE IS DISABLED;
   ```

   Row caching is also disabled if one of the conditions described in Section A.2 becomes false.

   When row caching is disabled, all previously created and assigned row caches remain in existence for future use when row caching is enabled again.

   The database must be closed when you enable or disable row caching.

2. The RMU/SET command allows you to enable or disable row caching using an unjournaled operation. This is needed to disable row caches if you have system tables mapped to row caches and you need to perform SQL operations that require exclusive database access.

   ```
   RMU/SET/ROW_CACHE[/DISABLED|/ENABLED] database_name
   ```

   For example, adding a row cache to a database requires exclusive database access. Execute this command before adding a new row cache using SQL then re-enable row caching.

3. The RMU/OPEN/ROW_CACHE=DISABLED command is used to keep row cache enabled in the database but not used for the duration of the open. This is necessary in order to set up row caching in a Hot Standby environment. Row caching is not allowed to be active on the standby database. Therefore, this command should be issued on the standby system to open the database without activating row caching.

A.4.2 Specifying Checkpointing and Sweeping Options

The following sections provide guidelines for specifying checkpointing and sweeping options.

A.4.2.1 Choosing the Checkpoint Source and Target Options

For greatest flexibility, provide each row cache with its own checkpoint source and target options as follows:

- The source rows to read

  This determines which source rows in the cache to write back to disk. Only updated rows or all rows can be selected. By default, only updated rows are selected.

- The target location to write the rows

This determines whether the source rows are written back to the database pages or written out to a separate row cache backing file.

You can specify the target location using the following parameters of the ADD, ALTER, and CREATE CACHE clauses. Note that you cannot specify that all rows are checkpointed to the database.

- CHECKPOINT UPDATED ROWS TO BACKING FILE
- CHECKPOINT UPDATED ROWS TO DATABASE
- CHECKPOINT ALL ROWS TO BACKING FILE

The following table lists the advantages and disadvantages of each checkpoint target:

Table A–2 Checkpoint Target Options

	Advantages	**Disadvantages**
Checkpoint to Database		
	Does not require any more disk space.	Is slower due to contention for database page buffers.
	Simpler to understand because it uses the traditional database page buffers.	Upon node failure, the row cache is not re-populated.
	Unmarks slots in the row cache so they can be reused for other rows.	Greater conflict with other users since row and page locks are maintained. The row cache server (RCS) process does not respond to requests to release row or page locks
	Writing back to database pages reclaims space on database pages from erased or modified rows that have been reduced in size.	
Checkpoint to Backing File		
	Can checkpoint all rows allowing a way to repopulate row caches that are predominantly read-only while recovering from a node failure.	Requires extra disk space to create two backing files per cache.
	Faster at writing sequential I/O operations to backing file.	Only used for node failure protection.
	Can be placed on different spindles so that other database I/O activity will not be impacted.	Marked rows tend to stay marked. By definition, rows in a row cache are only unmarked when they are written back to the database.
	Used upon node failure to repopulate the row cache.	Space on the database pages resulting from erased rows and modified rows that are reduced in size is not reclaimed.

A.4.2.2 Choosing the Checkpoint Interval

You must specify a checkpoint interval in the following way: use the CHECKPOINT TIMED EVERY s SECONDS parameter of the ROW CACHE IS ENABLED clause. This checkpoint parameter applies to the RCS process only.

This value can be overridden by the RDM$BIND_CKPT_TIME logical (this logical is also used to override the FAST COMMIT checkpoint interval). If nothing is specified, Oracle Rdb uses a default checkpoint interval of 15 minutes.

A.4.2.3 Specifying Sweeping Parameters

You set the number of updated rows that will be swept by using the NUMBER OF SWEEP ROWS IS parameter of the ADD, ALTER, and CREATE CACHE clause.

```
SQL> ALTER DATABASE FILENAME INVENTORY
cont> ALTER CACHE PARTS
cont> ROW LENGTH IS 104 BYTES
cont> CACHE SIZE IS 2000 ROWS
cont> CHECKPOINT ALL ROWS TO BACKING FILE
cont> NUMBER OF SWEEP ROWS IS 200;
```

A row in a row cache cannot be reused if it is marked (modified) or if its reference count is greater than zero. In the latter case, one or more users have a reference to this row in their row cache working sets. The RCS sweep operation tries to eliminate these restrictions from rows in the row cache so these rows can be reused to insert new rows.

The RCS process writes committed modified rows back to the database, up to a maximum of the NUMBER OF SWEEP ROWS defined for the row cache. It is important that this value be set properly so that when a sweep is initiated, the RCS process clears out enough slots to allow sufficient insertion activity before another sweep operation is necessary. Typically, a value of 10 percent to 30 percent of the size of the row cache would be sufficient. Make sure that the sweep count is larger than the value of the row cache's reserved count, specified by the NUMBER OF RESERVED ROWS IS N clause.

You can override the row cache's defined sweep count value by defining the RDM$BIND_RCS_SWEEP_COUNT logical name. Note, however, the value of this logical name applies to all row caches.

During a sweep operation, the RCS process may also initiate a dialogue with current users to reset the reference counts of the rows in the cache. The RCS process will only do this during a sweep operation if the number of database recovery processes since the last sweep operation of this row cache has exceeded the number specified by the RDM$BIND_RCS_CLEAR_GRICS_DBR_CNT logical name. Only processes that have abnormally terminated fail to clean up their reference counts normally.

An RCS sweep operation is triggered when a row cache is considered "clogged". A row cache is considered clogged when a user fails to find any available slots in which to insert rows. Even after a row cache is considered full, a user may still be able to insert rows into that row cache if the user still has reserved slots to use.

The RCS process clears the clogged flag if the sweep operation was successful in opening up some slots. The clogged flag can also become clear during a checkpoint operation if the RCS process has detected row cache entries with zero reference counts. This will only happen if the clogged flag stays set for three consecutive checkpoint operations.

A.4.2.4 Specifying the Size and Location of the Cache Backing File

When allocating the size of the cache backing (.RDC) files, consider the following:

- Whether all rows or only marked rows will be checkpointed
- The amount of update activity in the row cache
- Whether you want to create new backing files on each database open or re-use existing backing files

If you want Oracle Rdb to automatically rebuild an entire row cache in memory after a node failure, then define the row cache to checkpoint all rows to a cache backing file. If you want Oracle Rdb to repopulate the row cache with only the rows that were modified at the time, then define the row cache to checkpoint only updated rows to the cache backing file.

The decision you make determines how to size the cache backing files.

If all rows are to be checkpointed, use the following formula to determine the number of blocks to allocate for the cache backing file.

```
     Number of
     blocks    = (slot count * (row length + 40)) / 512 bytes per block
```

If only the updated rows are to be written to the backing file, use the following formula to allocate the backing file, based on the estimated number of updated rows in the row cache.

```
     Number of
     blocks    = (# of updated rows * (row length + 40)) / 512 bytes per block
```

You can overwrite the allocation specified in the row cache definition with the RDM$BIND_CKPT_FILE_SIZE system logical name. This specifies the percentage of the row cache size to allocate for the backing file. The default is 40 percent.

```
     Number of
     blocks    = (0.40 * slot count * (row length + 40)) / 512 bytes per block
```

When checkpointing to backing files, Oracle Rdb needs two backing files for each cache. One is used for the last checkpoint (committed rows), and the other is for the current checkpoint. Make sure there is enough disk space for two backing files for each cache. By default, Oracle Rdb deletes the backing files upon successful database shutdown and recreates them when the database is reopened. If you prefer, you can tell Oracle Rdb to save the backing files and re-use them on the subsequent database open by defining the system logical RDM$BIND_RCS_KEEP_BACKING_FILES to “1”.

If you are checkpointing a row cache to the database, you do not need to specify an allocation or location for the cache backing file. Oracle Rdb will ignore these clauses.

If you have a read-only cache, specify 1 block for the size of the cache backing file as follows:

```
SQL> ALTER DATABASE FILENAME MF_PERSONNEL
cont> ADD CACHE RCACHE_2
cont> LOCATION IS WORK$DISK1:[RCS]
cont> ALLOCATION IS 1 BLOCK;
```

A.4.3 Controlling What is Cached in Memory

The ROW REPLACEMENT parameter of the ADD, ALTER, and CREATE CACHE clause gives you some control over what happens when a row cache becomes full. If row replacement is enabled for a particular row cache, new rows will replace the oldest, unused, unmarked rows once the cache is full. If row replacement is disabled, new rows are not placed in the cache once the cache is full; they will always be retrieved from disk.

When you use the ROW REPLACEMENT IS DISABLED parameter, the data that was memory resident stays that way and therefore all subsequent reads occur from memory rather than disk.

You can increase performance by making the following types of rows memory resident.

- Nonleaf nodes of a B-tree index

  Be sure to account for the nodes splitting when you specify the size for the row cache. If a parent node splits and there is no room in the cache for the new node, the new node will not be held in memory.

- Data that is primarily read-only

  Data that does not change very often, such as dimension tables in a data warehouse environment, is a good candidate for keeping resident in memory.

- Data that is update-intensive; when the entire table can fit in the cache

  Oracle Rdb optimizes access when the cache is defined with row replacement disabled.

Enabling row replacement is beneficial when access patterns of a table are random. This ensures that the most frequently accessed rows remain in memory. Often, there may not be enough physical memory to cache an entire table, so caching the most frequently used rows can improve performance.

A.4.3.1 Row Replacement Strategy

Global and local buffers use the least-recently used (LRU) replacement strategy for database pages. Row caching uses a modified form of the LRU replacement strategy. Each database user can protect the last 10 rows they accessed. This group of rows is referred to as a **working set**. Rows that belong to a working set are considered to be **referenced** and are not eligible for row replacement. Any row that is in a cache and is not part of a working set is considered an **unreferenced** row. The unreferenced rows are eligible for replacement if they are not marked.

A.4.3.2 Inserting Rows into a Cache

Each user process requests rows from the database. A user process, which reads a row from a storage area, tries to insert the row into the cache (if it is not already there). If a slot is available, the requested row is stored in the cache, if it fits. If no more slots are available in the cache, one of the following happens:

- If ROW REPLACEMENT IS ENABLED, and an unmarked, unreferenced row can be found, that row is replaced by the new row. Oracle Rdb chooses the unreferenced row randomly.
- If ROW REPLACEMENT IS DISABLED, the row is not stored in the cache. This means that when the cache fills, it will not accept new rows. Reserved slots, however, can still be used.

You can prevent individual processes from inserting new rows into any Oracle Rdb row cache by defining the process logical RDM$BIND_RCACHE_INSERT_ENABLED to “0”. When defined, a process can only use what already exists in the row caches; the process cannot insert a row into a row cache. This option is useful if, for example, you want to keep nightly batch processes that perform large reporting functions from filling up row caches that are also used by the more important, daily, on-line transaction processing servers.

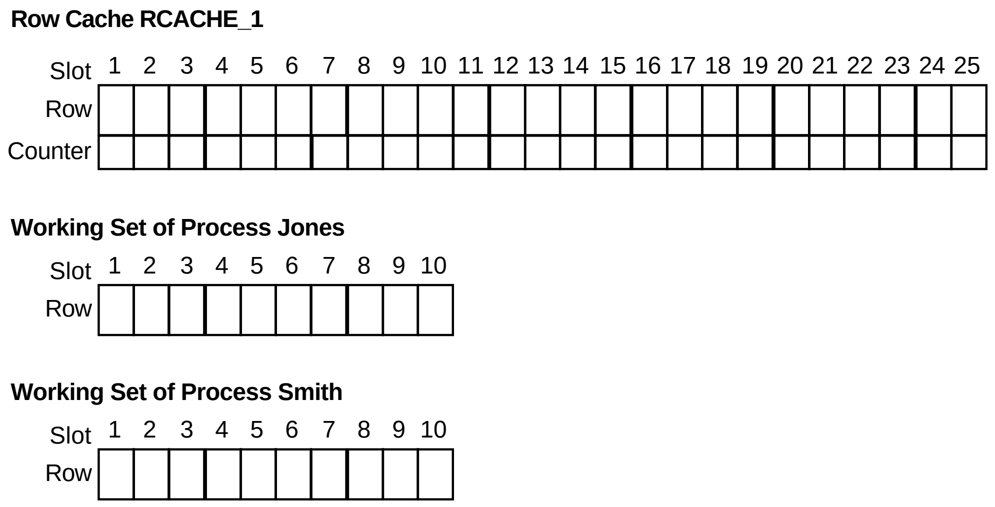
If system usage is lighter at night, you may want to preload row caches so that the data is available in memory during the day when database activity is at its peak.

The remainder of this section illustrates how Oracle Rdb inserts rows into a cache.

The example makes the following assumptions:

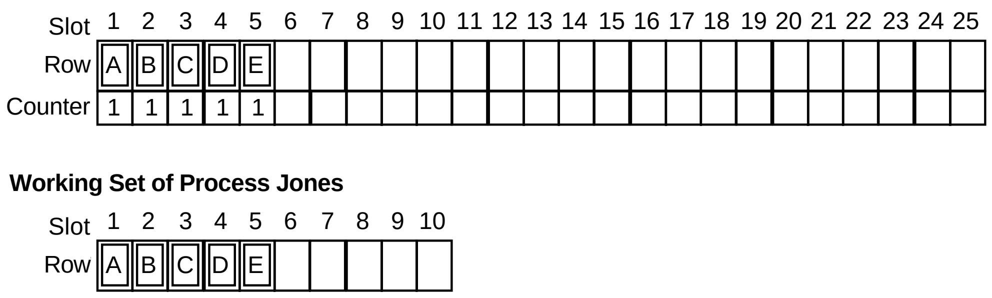
- Row caching is enabled.
- Row replacement is enabled.
- A row cache (RCACHE_1) has been created with 25 slots.
- Two processes (Jones and Smith) are attached to the database.
- The rows in the row cache are not modified.

The initial allocation is as follows:


Row Cache RCACHE_1
Slot 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
Row
Counter
Working Set of Process Jones
Slot 1 2 3 4 5 6 7 8 9 10
Row
Working Set of Process Smith
Slot 1 2 3 4 5 6 7 8 9 10
Row

NU−3614A−RA

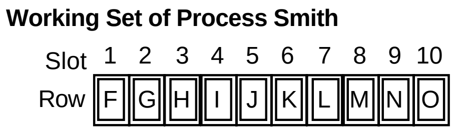
1. Process Jones executes a query that causes 5 rows to be read into the first 5 slots of the row cache.


Slot 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
Row A B C D E
Counter 1 1 1 1 1
Working Set of Process Jones
Slot 1 2 3 4 5 6 7 8 9 10
Row A B C D E

NU−3615A−RA

Each row slot has a working set counter associated with it. The working set counter indicates whether the row belongs to a working set. A positive value indicates that the row belongs to a working set. If a row belongs to a working set, it is not eligible for row replacement.

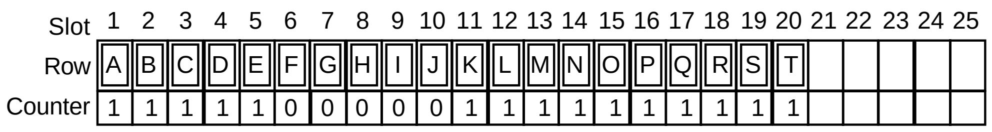
2. Process Smith requests 15 rows from the database. The first 10 rows requested go into Smith's working set as follows:



NU−3616A−RA

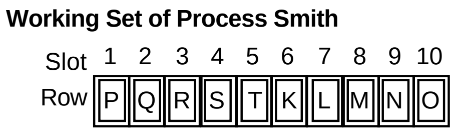
Process Smith's working set has exactly 10 slots, and all 10 are being used. The least recently used row is replaced by the eleventh row that Process Smith reads into the cache. Rows 12 through 15 also overwrite the contents of slots 2 through 5 respectively.

After the 15 rows are read into the cache, the cache appears as follows:


Slot 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
Row A B C D E F G H I J K L M N O P Q R S T
Counter 1 1 1 1 1 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1

NU−3617A−RA

After the 15 rows are read into the cache, Process Smith's working set appears as follows:



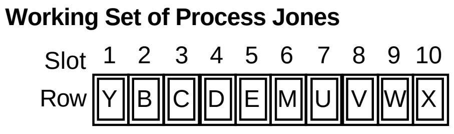
NU−3618A−RA

At this point, rows F, G, H, I, and J are unreferenced. They are in the cache but they do not belong to the working set of any process. Oracle Rdb sets the working set counter for an unreferenced row to zero. The unreferenced rows are eligible for replacement if they have not been modified and row replacement is enabled. Any process can read rows F, G, H, I, or J without executing an I/O operation. However, if a process requires a row that is not currently in the cache, one of the rows F, G, H, I, or J is replaced with the new row.

Each slot in the row cache contains a modification flag. If the row has been modified, but not yet flushed to disk, it is considered to be **dirty**. Dirty rows are not candidates for row replacement either. Modified rows are written to disk by the row cache server (RCS) process. See Section A.4.2.1 for more information.

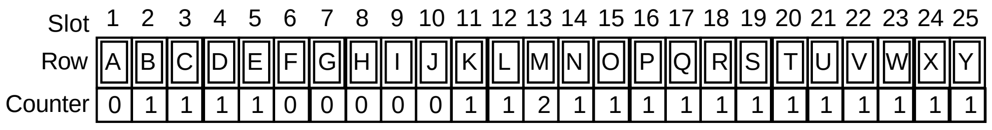
3. Process Jones requests 7 more rows: M, U, V, W, X, Y, and Z. Jones can read row M without performing any I/O because M is already in the cache. An additional slot does not get filled in the row cache, but row M is added to Process Jones' working set.

   Process Jones' working set now appears as follows:



NU−3619A−RA

   Rows U, V, W, X, and Y go into the remaining slots in the row cache and the row cache appears as follows:



NU−3620A−RA

   Note that the working set counter for slot 13 indicates that row M is in two working sets. This indicates that two processes are accessing the same row. The number of processes sharing a particular slot is known as the **share count**.

   At this point, the cache is full. If row replacement were disabled for the row cache, then row Z could not be inserted. However, in this example, row replacement is enabled, and there is an unreferenced slot. Therefore, Oracle Rdb will choose an unreferenced slot to make room for the new row, Z. (In this example, the unreferenced slots are A, F, G, H, I, and J.)

A.5 Examining Row Cache Information

You can display the attributes using the SHOW CACHE statement as in the following example:

```
SQL> SHOW CACHE PARTS;
     PARTS
           Cache Size:         204 rows
           Row Length:         104 bytes
           Row Replacement:    Enabled
           Shared Memory:      Process
           Large Memory:       Disabled
           Window Count:       100
           Reserved Rows:      20
           Sweep Rows:         1004
           Allocation:         100 blocks
           Extent:             100 blocks
```

You can also use the RMU Dump command with the Header qualifier to display row cache information, as in the following example:

Example A–4 Row Cache Parameters

```
$ RMU/DUMP/HEADER INVENTORY
   .
   .
   .
    Row Caches...          1
      - Active row cache count is 4
      - Reserved row cache count is 20
      - Checkpoint information
          Time interval is 10 seconds
          Default source is updated rows
          Default target is backing file
          Default backing file directory is "DISK1:[RDB]"
   .
   .
   .
Row cache "PARTS"
    Cache ID number is 4 2
    Allocation...   3
      - Row slot count is 204
      - Maximum row size allowed in cache is 104 bytes
      - Working set count is 10
      - Maximum slot reservation count is 20
      - Row replacement is enabled
    Sweeping...   4
      - Sweep row count is 1004
      - Maximum batch I/O count is 0
    Checkpointing...  5
      - Source is updated rows (database default)
      - Target is backing file (database default)
      - No checkpoint information available
      - Checkpoint sequence is 0
    Files...   6
      - Default cache file directory is "DISK1:[RDB]"
      - File allocation is 100 blocks
      - File extension is 100 blocks
    Hashing...   7
      - Hash value for logical area DBIDs is 211
      - Hash value for page numbers is 11
    Shared Memory...  8
      - System space memory is disabled
      - Large memory is disabled
      - Large memory window count is 100
    Cache-size in different sections of memory... 9
      - Without VLM, process or system memory requirement
        is 309760 bytes
      - With VLM enabled...
          - Process or system memory requirement is 38768 bytes
          - Physical memory requirement is 280000 bytes
          - VLM Virtual memory address space requirement is
     approximately 102400 bytes
   .
   .
   .
```

The following callouts identify the parameters in Example A–4:

1 Row Caches . . .

- Active row cache count is 4

This specifies the number of row caches currently defined in this database.

- Reserved row cache count is 20

  This specifies the number of slots that are available in the database. The cache slots are reserved with the RESERVE n CACHE SLOTS parameter of the ALTER or CREATE DATABASE statements.

- Checkpoint information

  This displays database-level checkpoint information specified using parameters of the ADD, ALTER, or CREATE CACHE clauses.

  – Time interval is 10 seconds

    A checkpoint is one full pass through all active row caches, attempting to write all or just marked rows back to their respective storage areas or the backing file. The time interval is set with the CHECKPOINT TIMED EVERY s SECONDS parameter.

  – Default source is updated rows

    Only updated rows are written to the backing file or back to the database storage areas.

  – Default target is backing file

    Specifies that the default target for the checkpoint is the backing file and not the database. This is the default target when the CHECKPOINT UPDATED ROWS parameter is not set.

  – Default backing file directory is “DISK1:[RDB]”.

    The default cache file directory is the directory where Oracle Rdb places the cache backing store files. If you do not explicitly include a directory specification, Oracle Rdb will place the backing file in the directory where the database root file is stored.

2 Cache ID number is

Oracle Rdb assigns an ID to each defined row cache in the database.

3 Allocation . . .

- Row slot count is 204

  This is specified with the CACHE SIZE IS n ROWS parameter.

- Maximum row size allowed in cache is 104 bytes

  This is specified with the ROW LENGTH IS n BYTES parameter.

- Working set count is 10

  This is the number of “in use” rows that are not eligible for row replacement.

- Maximum slot reservation count is 20

  This is specified with the NUMBER OF RESERVED ROWS parameter. The default value is 20 rows.

  The number of reserved rows indicates how many slots in the cache Oracle Rdb will reserve for each process. Reserving many rows minimizes row cache locking while rows are inserted into the cache.

The number of reserved rows parameter is also used when searching for available slots in a row cache. The entire row cache is not searched on the initial pass. This parameter is used as the maximum number of rows that are searched for a free slot. If at least one free slot is found, the insert operation can proceed. If no free slots are found in this initial search, Oracle Rdb will continue searching through the cache until it finds a free slot.

- Row replacement is enabled

  This is specified with the ROW REPLACEMENT parameter. Row replacement is enabled by default.

4 Sweeping . . .

- Sweep row count is

  Sets the number of marked rows that will be swept back to the database or backing file when the row cache is full and a user attempts to find an empty slot.

5 Checkpointing . . .

- Source is updated rows (database default)

  The source of updated rows is the same as the database default.

- Target is backing file (database default)

  The target for marked rows is the database default.

6 Files . . .

- Default cache file directory is "DISK1:[RDB]"

  The LOCATION parameter specifies a directory specification for the cache backing store file. Oracle Rdb writes to the cache backing store file when the RCS process checkpoints. Oracle Rdb automatically generates a file name with a file extension of .rdc. The default location for the cache backing store file is the directory where the database root file is stored.

  The LOCATION parameter can be specified at the database level or at the row cache level. If you include the LOCATION parameter in the ADD CACHE or CREATE CACHE clauses of the CREATE or ALTER DATABASE statements, the directory you specify becomes the default directory location for all the row caches that are defined for the database. You can, however, override the default directory location for individual row caches by specifying the LOCATION parameter in the row cache definition.

- File allocation is 100 blocks

  The ALLOCATION parameter specifies the initial size of the cache backing file. The default allocation is 40 percent of the cache size. The cache size is determined by multiplying the number of rows in the cache by the row length.

- File extension is 100 blocks

  The EXTENT parameter specifies the number of pages by which the cache backing store file can be extended after the initial allocation has been reached. The default extent is 127 multiplied by the number of rows in the cache.

7 Hashing . . .

- Hash value for logical area DBIDs is 211
- Hash value for page numbers is 11

  The hash values are used by Oracle Rdb to fine-tune the distribution of hash table queues in the row cache.

8 Shared Memory . . .

- System space memory is disabled

  This is specified with the SHARED MEMORY parameter. This specifies whether Oracle Rdb creates the row cache in shared memory. The row cache is created in a process global section (OpenVMS) or in a shared memory partition (Digital UNIX) by default.
- Large memory is disabled

  This is specified with the LARGE MEMORY parameter. This specifies whether Oracle Rdb creates the row cache in physical memory. Large memory is disabled by default.
- Large memory window count is 100

  This is specified with the WINDOW COUNT parameter. The default value is 100 windows. The WINDOW COUNT specifies how many locations of the physical memory are mapped to each user's private window in virtual address space.

9 Cache-size in different sections of memory . . .

- Without VLM, process or system memory requirement is 309760 bytes

  When the cache is created in a process global section or system space buffer and VLM is not enabled, this is the memory requirement.
- With VLM enabled . . .
  - Process or system memory requirement is 38768 bytes

    When VLM is enabled and the cache is created in a process global section or system space buffer, this is the memory requirement.
  - Physical memory requirement is 280000 bytes

    The actual cached data requires this space in VLM.
  - VLM Virtual memory address space is approximately 102400 bytes

    This is the address space used by the virtual memory windows.

A.5.1 RMU Show Statistics Screens and Row Caching

The RMU Show Statistics command displays much information regarding row caches. The following are titles of some of the screens that can be displayed regarding row cache:

- Summary Cache Statistics
- Summary Cache Unmark Statistics
- Row Cache (One Cache)
- Row Cache (One Field)
- Row Cache Utilization

- Hot Row Information
- Row Cache Status
- Row Cache Queue Length
- Row Length Distribution
- RCS Statistics
- Row Cache Dashboard
- RCS Dashboard
- Per-Process Row Cache Dashboard

A.6 Examples

This section includes some practical examples on using the row cache feature of Oracle Rdb.

A.6.1 Loading a Logical Area Cache

Use the following steps to place an entire table in a row cache:

1. Determine how many rows are in the table.

```
SQL> SELECT COUNT(*) FROM EMPLOYEES;
         100
1 row selected
```

2. Create a logical cache large enough to hold to the table.

   Use the table name as the name of the cache to create the logical cache. Oracle Rdb will determine the row length from the table.

```
SQL> ALTER DATABASE FILENAME MF_PERSONNEL
cont> ADD CACHE EMPLOYEES
cont> CACHE SIZE IS 100 ROWS;
```

3. Cause Rdb to sort the table by an indexed field.

   This causes rows to be read by DBKEY after the sort is complete.

```
SQL> SELECT * FROM EMPLOYEES ORDER BY EMPLOYEE_ID;
 EMPLOYEE_ID   LAST_NAME        FIRST_NAME   MIDDLE_INITIAL
   ADDRESS_DATA_1              ADDRESS_DATA_2         CITY
      STATE   POSTAL_CODE   SEX    BIRTHDAY      STATUS_CODE
  00197         Danzig           Chris        NULL
    136 Beaver Brook Circle                          Acworth
       NH      03601         F      21-Jun-1939   1
   .
   .
   .
```

A.6.2 Caching Database Metadata

Because metadata is frequently accessed, you may want to cache some or all of your database’s metadata. You can map the entire contents of the RDB$SYSTEM storage area to a physical area row cache. Alternatively, you can map certain system tables, such as RDB$RELATIONS and RDB$INDICES, into separate logical area row caches.

To do this, follow these steps.

1. Use the RMU/DUMP/AREA command to display the contents of the storage area. (Note that the RMU Dump command output uses the term records to refer to rows.)

```
$RMU/DUMP/AREA=RDB$SYSTEM/OUT=RMU_DUMP_1.OUT MF_PERSONNEL
$SEARCH/STATISTICS RMU_DUMP_1.OUT "RECORD LENGTH", "STATIC_DATA"
                            00A2  0050  record length 162 bytes
                            00E8  008B  record length 232 bytes
                            00C4  00C6  record length 196 bytes
                            00E4  0101  record length 228 bytes
                            0088  013C  record length 136 bytes
                            023C  0177  record length 572 bytes
                            0220  01B2  record length 544 bytes
                            030C  01ED  record length 780 bytes
   .
   .
   .
Files searched:                1      Buffered I/O count:      100
Records searched:          62260      Direct I/O count:        441
Characters searched:     3459752      Page faults:              20
Records matched:              96      Elapsed CPU time:  0 00:00:01.63
Lines printed:                96      Elapsed time:      0 00:00:02.83
```

2. Determine the row length and slot count.

   Keep in mind that other structures may be stored in this area because it can be specified as the default storage area for Oracle Rdb.

3. Add the physical cache and assign it to the RDB$SYSTEM storage area.

   In the following example, row length has been rounded up and the cache size has been increased to allow for future growth.

```
SQL> ALTER DATABASE FILENAME MF_PERSONNEL
cont> ADD CACHE RDB_SYSTEM_CACHE
cont> CACHE SIZE IS 9000 ROWS
cont> ROW LENGTH IS 800 BYTES;
SQL> ALTER DATABASE FILENAME MF_PERSONNEL
cont> ALTER STORAGE AREA RDB$SYSTEM
cont> CACHE USING RDB_SYSTEM_CACHE;
```

4. Or, add the logical area caches to the Rdb system tables of interest.

```
SQL> ALTER DATABASE FILENAME MF_PERSONNEL
cont>     ADD CACHE RDB$RELATIONS
cont>         CACHE SIZE IS 1000 ROWS
cont>         ROW LENGTH IS 500 BYTES
cont>     ADD CACHE RDB$INDICES
cont>         CACHE SIZE IS 2000 ROWS
cont>         ROW LENGTH IS 500 BYTES;
```

When caching metadata, you will experience conflicts when executing database operations through SQL that require exclusive database access. For example, adding new row caches or dropping existing ones requires exclusive database access. When the SQL command is parsed, the Oracle Rdb system tables are queried. This access to the system tables creates the row caches and causes the RCS process to come up to manage those row caches. As a result, the database now has another “user”, the RCS process. This causes the exclusive database operation to fail.

To resolve this, you must first turn off row caching temporarily using the RMU Set command specifying the Row_Cache and Disabled qualifiers. Then, perform the SQL operation that requires exclusive database access. Finally, re-enable row caching using the RMU Set command with the Row_Cache and Enabled qualifiers.

A.6.3 Caching a Sorted Index

To cache a sorted index, use the following steps:

1. Display the number of index nodes using the RMU Analyze Index command. (Note that the RMU Analyze command uses the term records to refer to rows.)

   ```
   $RMU/ANALYZE/INDEX MF_PERSONNEL EMP_LAST_NAME

    Index EMP_LAST_NAME for relation EMPLOYEES duplicates allowed
     Max Level: 2, Nodes: 8, Used/Avail: 1625/3184 (51%), Keys: 90, Records: 67
          Duplicate nodes: 16, Used/Avail: 264/312 (85%), Keys: 16, Records: 33
   ```

2. Count the number of nodes and duplicate nodes.
3. Allocate slots based on the number of nodes currently used and allow for future growth.

   In this example, allocating 28 slots would be reasonable.
4. Determine node and duplicate node size. Sorted indexes with duplicates should be sized at 430 bytes rounded up to the next 4-byte interval.
5. Create a logical cache for the sorted index.

   ```
   SQL> ALTER DATABASE FILENAME MF_PERSONNEL
   cont> ADD CACHE EMP_LAST_NAME
   cont> ROW LENGTH IS 440 BYTES
   cont> CACHE SIZE IS 28 ROWS;
   ```

B
Row Cache Statements

B.1 ALTER DATABASE Statement

B.1.1 Overview

Alters a database in any of the following ways:

- For single-file and multifile databases, the ALTER DATABASE statement changes the characteristics of the database root file.

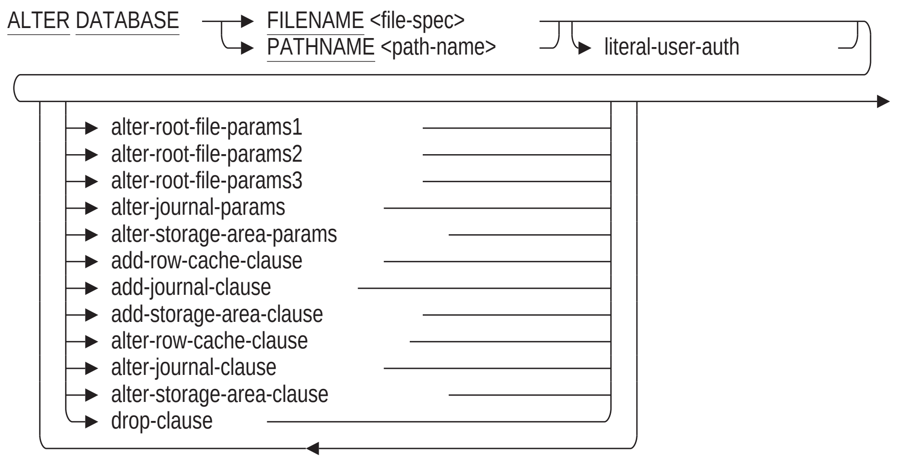
  The ALTER DATABASE statement lets you override certain characteristics specified in the database root file parameters of the CREATE DATABASE statement, such as whether or not a snapshot file is disabled. In addition, ALTER DATABASE lets you control other characteristics you cannot specify in the CREATE DATABASE database root file parameters, such as whether or not after-image journaling is enabled.
- For single-file and multifile databases, the ALTER DATABASE statement changes the storage area parameters.
- For multifile databases *only*, the ALTER DATABASE statement adds, alters, or deletes storage areas.

B.1.2 Environment

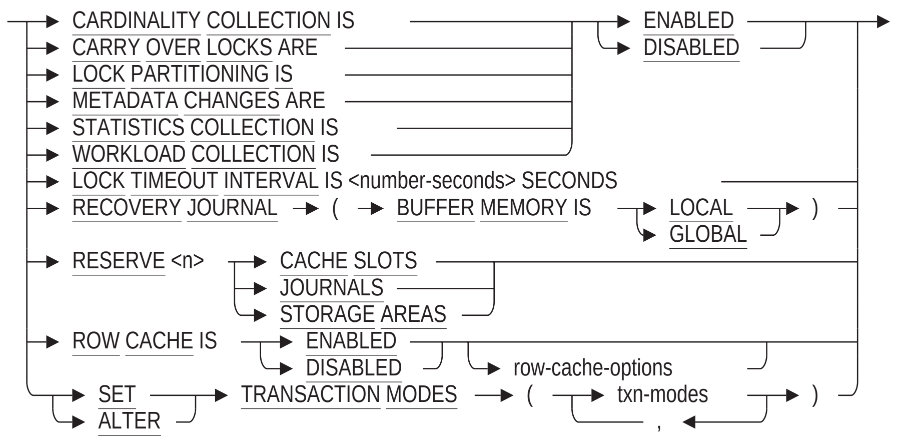
You can use the ALTER DATABASE statement:

- In interactive SQL
- Embedded in host language programs to be precompiled
- As part of a procedure in an SQL module
- In dynamic SQL as a statement to be dynamically executed

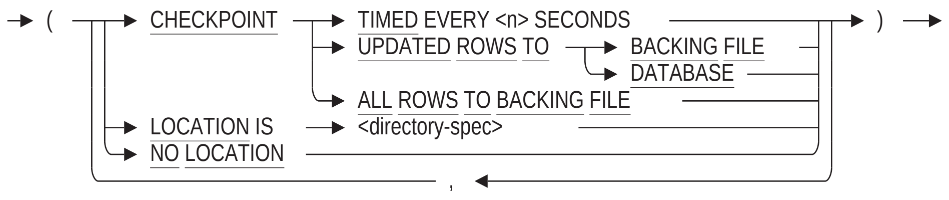
B.1.3 Format



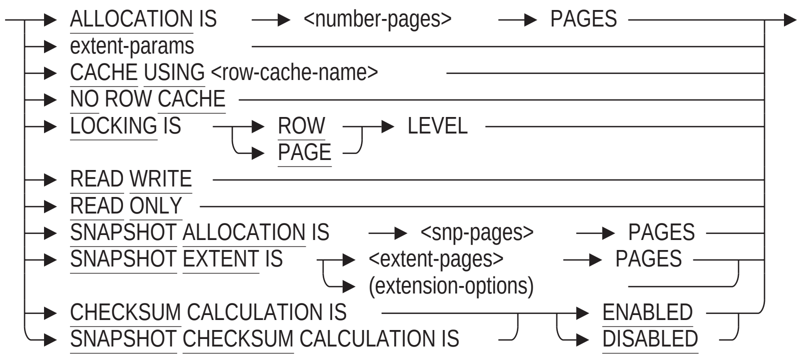
alter-root-file-params2 =



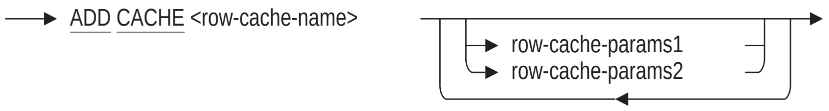
row-cache-options =



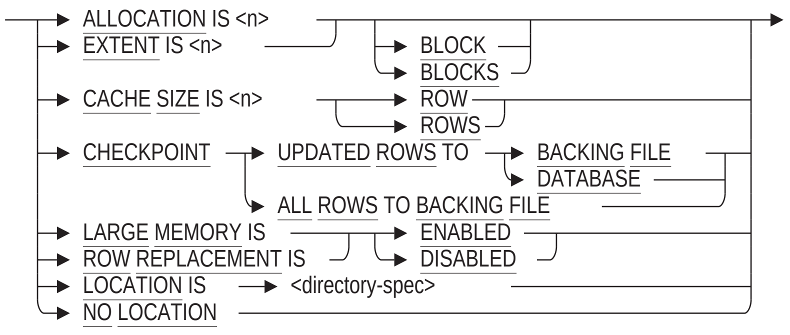
alter-storage-area-params =



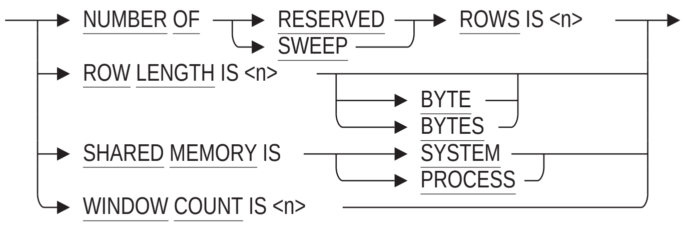
add-row-cache-clause =



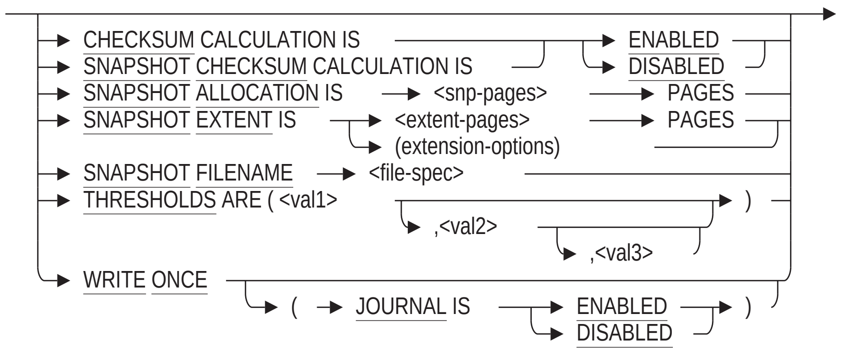
row-cache-params1 =



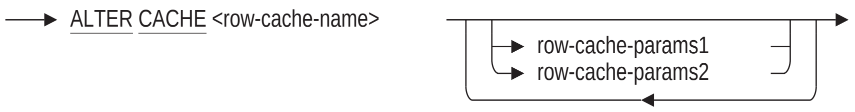
row-cache-params2 =



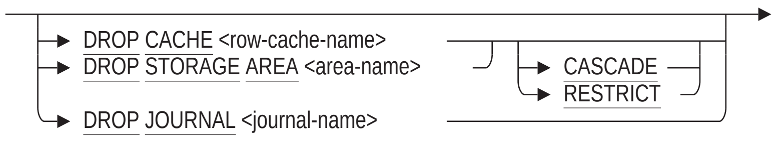
storage-area-params-2 =



alter-row-cache-clause =



drop-clause =


DROP CACHE <row-cache-name>
DROP STORAGE AREA <area-name>
CASCADE
RESTRICT
DROP JOURNAL <journal-name>

B.1.4 Arguments

B.1.4.1 RECOVERY JOURNAL (BUFFER MEMORY IS {LOCAL | GLOBAL})

The RUJ buffers used by each process are normally allocated in local virtual memory. With the introduction of ROW CACHE, these buffers can now be assigned to a shared global section (GLOBAL memory) so that the recovery process can process this in memory buffer and possibly avoid a disk access.

This buffer memory can be defined a GLOBAL to improve ROW CACHE performance for recovery. If ROW CACHE is DISABLED then buffer memory is always LOCAL.

B.1.4.2 RESERVE n CACHE SLOTS

Specifies the number of row caches for which slots are reserved in the database.

You can use the RESERVE CACHE SLOTS clause to reserve slots in the database root file for future use by the ADD CACHE clause. Row caches can be added only if there are row cache slots available. Slots become available after a DROP CACHE clause or a RESERVE CACHE SLOTS clause.

The number of reserved slots for row cache cannot be decreased once the RESERVE clause is issued. If you reserve 10 slots and later reserve 5 slots, you have a total of 15 reserved slots for row caches.

Reserving row cache slots is an offline operation (requiring exclusive database access).

B.1.4.3 CACHE USING row-cache-name

Assigns the named row cache as the default physical row cache for all storage areas in the database. All rows stored in each storage area, whether they consist of table data, segmented string data, or special rows such as index nodes, are cached.

The row cache must exist before terminating the ALTER DATABASE statement.

Alter the database and storage area to assign a new physical area row cache to override the database default physical area row cache. Only one physical area row cache is allowed for each storage area.

You can have multiple row caches containing rows for a single storage area by defining logical area row caches, where the row cache name matches the name of a table or index.

If you do not specify the CACHE USING clause or the NO ROW CACHE clause, NO ROW CACHE is the default for the database.

B.1.4.4 NO ROW CACHE

Specifies that the database default is not to assign a row cache to all storage areas in the database. You cannot specify the NO ROW CACHE clause if you specify the CACHE USING clause.

Alter the storage area and name a row cache to override the database default. Only one row cache is allowed for each storage area.

If you do not specify the CACHE USING clause or the NO ROW CACHE clause, NO ROW CACHE is the default for the database.

B.1.4.5 ROW CACHE IS ENABLED/ROW CACHE IS DISABLED

Specifies whether or not you want Oracle Rdb to enable the row caching feature.

Enabling cache support does not affect database operations until a cache is created and assigned to one or more storage areas.

When the row caching feature is disabled, all previously created and assigned caches remain in existence for future use when the row caching feature is enabled.

Enabling and disabling the row cache feature is an offline operation (requiring exclusive database access).

B.1.4.5.1 CHECKPOINT TIMED EVERY N SECONDS Specifies the frequency with which the RCS process checkpoints the contents of the row caches back to disk. **The RCS process does not use the checkpoint frequency options of the FAST COMMIT clause.**

The frequency of RCS checkpointing is important in determining how much of an AIJ file must be read during a REDO operation following a node failure. It also affects the frequency that marked records get flushed back to the database, for those row caches that checkpoint to the database. The default is every 15 minutes (900 seconds).

B.1.4.5.2 CHECKPOINT ALL ROWS TO BACKING FILE/ CHECKPOINT UPDATED ROWS TO BACKING FILE/ CHECKPOINT UPDATED ROWS TO DATABASE Specifies the default source and target for checkpoint operations for all row caches. If ALL ROWS is specified, then the source records written during each checkpoint operation are both the modified and the unmodified rows in a row cache. If UPDATED ROWS is specified, then just the modified rows in a row cache are checkpointed each time.

If the target of the checkpoint operation is BACKING FILE, then the RCS process writes the source row cache entries to the backing (.rdc) files. The row cache LOCATION, ALLOCATION, and EXTENT clauses are used to create the backing files. Upon recovery from a node failure, the database recovery process is able to re-populate the in-memory row caches from the rows found in the backing files.

If the target is DATABASE, then the target rows (only UPDATED ROWS is allowed) are written back to the database. The row cache LOCATION, ALLOCATION, and EXTENT clauses are ignored. Upon recovery from a node failure, the database recovery process has no data on the contents of the row cache. Therefore, it does not re-populate the in-memory row caches.

The CHECKPOINT clause of the CREATE CACHE, ADD CACHE, or ALTER CACHE clause overrides this database level CHECKPOINT clause.

B.1.4.5.3 LOCATION IS directory-spec Specifies the name of the default backing store directory to which all row cache backing files are written. The database system generates a file name automatically (row-cache-name.rdc) for each row cache backing file it creates when the RCS process first starts up. Specify a device name and directory name only, enclosed within single quotation marks. By default, the location is the directory of the database root file.

The LOCATION clause of the CREATE CACHE, ADD CACHE, or ALTER CACHE clause overrides this location which is the default for the database.

B.1.4.5.4 NO LOCATION Removes the location previously specified in a LOCATION IS clause for the database for the row cache backing file. If you specify NO LOCATION, the row cache backing file location becomes the directory of the database root file.

The LOCATION clause of the CREATE CACHE, ADD CACHE, or ALTER CACHE clause overrides this location which is the default for the database.

B.1.4.6 ADD CACHE clause

Creates a new row cache.

B.1.4.6.1 ALLOCATION IS n BLOCK/ALLOCATION IS n BLOCKS Specifies the initial allocation of the row cache backing file (.rdc) to which cached rows are written during a checkpoint operation.

If the ALLOCATION clause is not specified, the default allocation in blocks is approximately 40 percent of the CACHE SIZE for this row cache.

This clause is ignored if the row cache is defined to checkpoint to the database.

B.1.4.6.2 CACHE SIZE IS n ROW/CACHE SIZE IS n ROWS Specifies the number of rows allocated to the row cache. As the row cache fills, rows more recently referenced are retained in the row cache while those not referenced recently are discarded. Adjusting the allocation of the row cache helps to retain important rows in memory. If not specified, the default is 1000 rows.

The product of the CACHE SIZE and the ROW LENGTH settings determines the amount of memory required for the row cache. (Some additional overhead and rounding up to page boundaries is performed by the database system.) The row cache is shared by all processes attached to the database.

B.1.4.6.3 CHECKPOINT ALL ROWS TO BACKING FILE/ CHECKPOINT UPDATED ROWS TO BACKING FILE/ CHECKPOINT UPDATED ROWS TO DATABASE Specifies the source and target for checkpoint operations for the row cache. If ALL ROWS is specified, then the source records written during each checkpoint operation are both the modified and the unmodified rows in the row cache. If UPDATED ROWS is specified, then just the modified rows in the row cache are checkpointed each time.

If the target of the checkpoint operation is BACKING FILE, then the RCS process writes the source row cache entries to the backing (.rdc) files. The row cache LOCATION, ALLOCATION, and EXTENT clauses are used to create the backing files. Upon recovery from a node failure, the database recovery process is able to re-populate the in-memory row caches from the rows found in the backing files.

If the target is DATABASE, then the target rows (only UPDATED ROWS is allowed) are written back to the database. The row cache LOCATION, ALLOCATION, and EXTENT clauses are ignored. Upon recovery from a node failure, the database recovery process has no data on the contents of the row cache. Therefore, it does not re-populate the in-memory row caches.

This CHECKPOINT clause overrides the database level CHECKPOINT clause.

B.1.4.6.4 EXTENT IS n BLOCK/EXTENT IS n BLOCKS Specifies the file extent size for the row cache backing file (.rdc).

If the EXTENT clause is not specified, the default number of blocks is CACHE SIZE * 127 for this cache.

This clause is ignored if the row cache is defined to checkpoint to the database.

B.1.4.6.5 LARGE MEMORY IS ENABLED/LARGE MEMORY IS DISABLED Specifies whether or not large memory is used to manage the row cache. Very large memory (VLM) allows Oracle Rdb to use as much physical memory as is available. It provides access to a large amount of physical memory through small virtual address windows.

Use LARGE MEMORY IS ENABLED only when both of the following are true:

- You have enabled row caching.
- You want to cache large amounts of data, but the cache does not fit in the virtual address space.

The default is DISABLED. See the Usage Notes for restrictions pertaining to the very large memory (VLM) feature.

B.1.4.6.6 LOCATION IS directory-spec Specifies the name of the default backing store directory to which all row cache backing files are written. The database system generates a file name automatically (row-cache-name.rdc) for each row cache backing file it creates when the RCS process first starts up. Specify a device name and directory name only, enclosed within single quotation marks. By default, the location is the directory of the database root file.

This LOCATION clause overrides a previously specified location at the database level.

This clause is ignored if the row cache is defined to checkpoint to the database.

B.1.4.6.7 NO LOCATION Removes the location previously specified in a LOCATION IS clause for the database for the row cache backing file. If you specify NO LOCATION, the row cache backing file location becomes the directory of the database root file.

This clause is ignored if the row cache is defined to checkpoint to the database.

B.1.4.6.8 NUMBER OF RESERVED ROWS IS n Specifies the maximum number of cache rows that each user can reserve. The default is 20 rows.

The number of reserved rows parameter is also used when searching for available slots in a row cache. The entire row cache is not searched on the initial pass. This parameter is used as the maximum number of rows that are searched for a free slot. If at least one free slot is found, the insert operation can proceed. If no free slots are found in this initial search, Oracle Rdb will continue searching through the cache until it finds a free slot.

B.1.4.6.9 NUMBER OF SWEEP ROWS IS n Specifies the number of modified cache rows that will be written back to the database to make space available in the cache for subsequent transactions to insert rows into the cache. It is recommended that users initially specify the number of sweep rows to be between ten and thirty percent of the total number of rows in the cache. Users should then monitor performance and adjust the number of sweep rows if necessary. The default setting is 3000 rows.

B.1.4.6.10 ROW LENGTH IS n BYTE/ROW LENGTH IS n BYTES Specifies the size of each row allocated to the row cache. Rows are not cached if they are longer than a row cache row. The ROW LENGTH is an aligned longword rounded up to the next multiple of 4 bytes.

If the ROW LENGTH clause is not specified, the default row length is 256 bytes.

B.1.4.6.11 ROW REPLACEMENT IS ENABLED/ROW REPLACEMENT IS DISABLED Specifies whether or not Oracle Rdb replaces rows in the cache. When the ROW REPLACEMENT IS ENABLED clause is used, rows are replaced when the row cache becomes full. When the ROW REPLACEMENT IS DISABLED clause is used, rows are not replaced when the cache is full. The type of row replacement policy depends upon the application requirements for each cache.

The default is ENABLED.

B.1.4.6.12 SHARED MEMORY IS SYSTEM/SHARED MEMORY IS PROCESS

Determines whether cache global sections are created in system space or process space. The default is SHARED MEMORY IS PROCESS.

When you use cache global sections created in the process space, you and other users share physical memory and the OpenVMS Alpha operating system maps a row cache to a private address space for each user. As a result, all users are limited by the free virtual address range and each use a percentage of memory in overhead. If many users are accessing the database, the overhead can be high.

When many users are accessing the database, consider using SHARED MEMORY IS SYSTEM. This gives users more physical memory because they share the system space of memory and there is none of the overhead associated with the process space of memory.

B.1.4.6.13 WINDOW COUNT IS n Specifies the number of virtual address windows used by the LARGE MEMORY clause.

The window is a view into the physical memory used to create the very large memory (VLM) information. Because the VLM size may be larger than that which can be addressed by a 32-bit pointer, you need to view the VLM information through small virtual address windows.

You can specify a positive integer in the range from 10 through 65535. The default is 100 windows.

B.1.4.7 ALTER CACHE row-cache-name

Alters existing row caches.

B.1.4.7.1 row-cache-params For information regarding the row-cache-params, see the descriptions under the ADD CACHE argument described earlier in this arguments list.

B.1.4.7.2 DROP CACHE row-cache-name CASCADE

B.1.4.7.3 DROP CACHE row-cache-name RESTRICT Deletes the specified row cache from the database.

If the mode is RESTRICT, an exception is raised if the row cache is assigned to a storage area.

If the mode is CASCADE, the row cache is removed from all referencing storage areas.

The default is RESTRICT if no mode is specified.

B.2 CREATE DATABASE

B.2.1 Overview

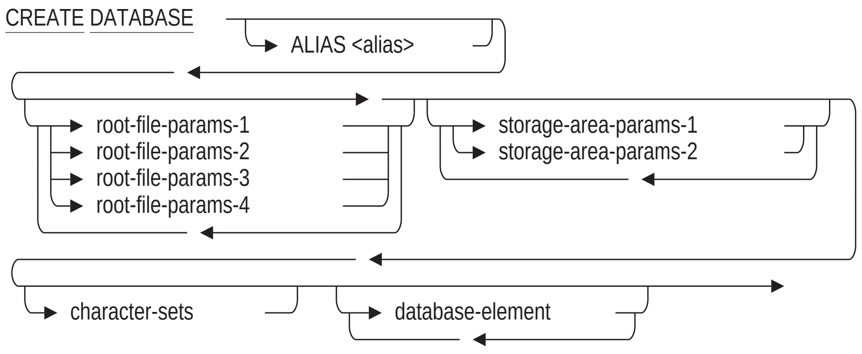
Creates database system files, metadata definitions, and user data that comprise a database. The CREATE DATABASE statement lets you specify in a single SQL statement all data and privilege definitions for a new database. (You can also add definitions to the database later.) For information about ways to ensure good performance and data consistency, see the *Oracle Rdb Guide to Database Performance and Tuning*.

B.2.2 Environment

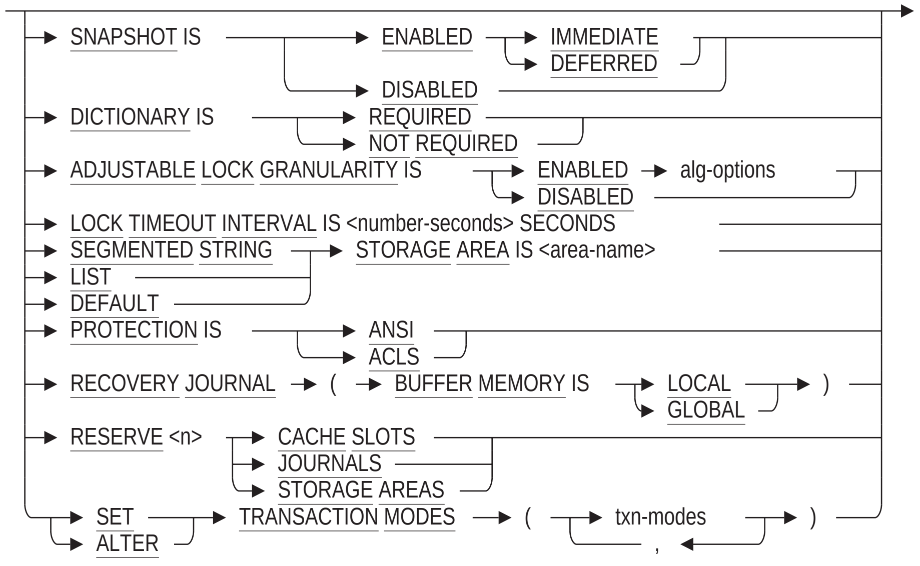
You can use the CREATE DATABASE statement:

- In interactive SQL
- Embedded in host language programs to be precompiled
- As part of a procedure in an SQL module
- In dynamic SQL as a statement to be dynamically executed

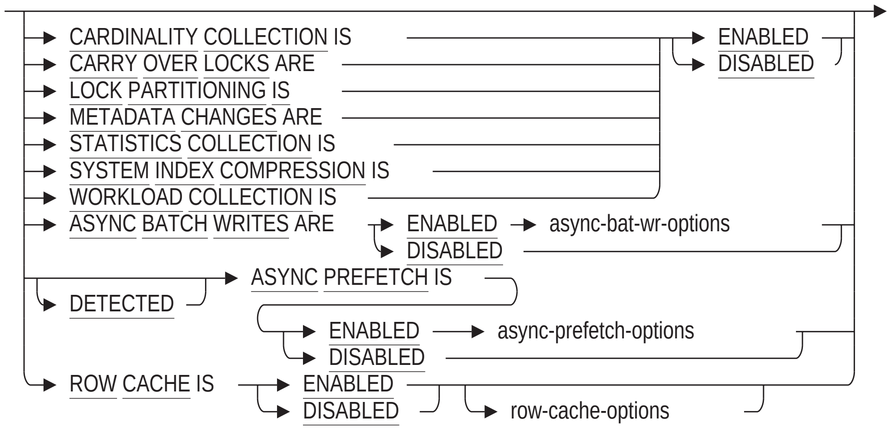
B.2.3 Format



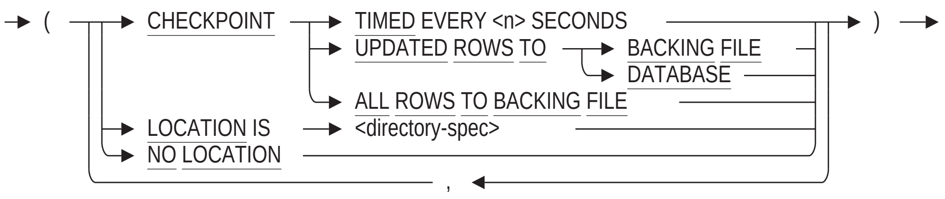
root-file-params-2 =



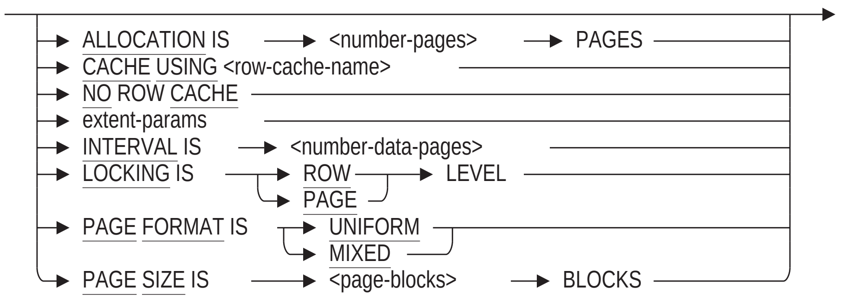
root-file-params-3 =



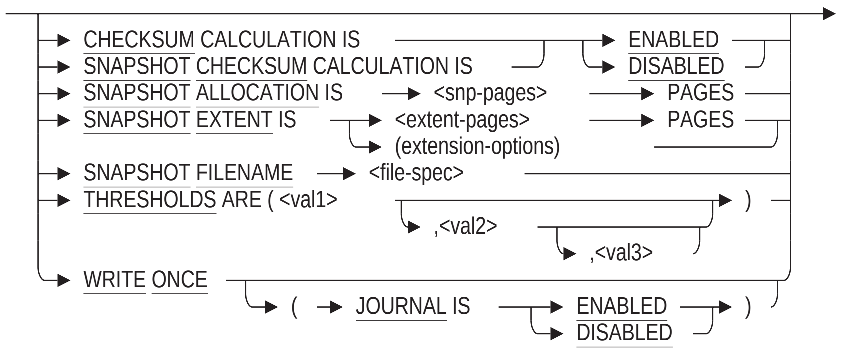
row-cache-options =



storage-area-params-1 =



storage-area-params-2 =



B.2.4 Arguments

B.2.4.1 RECOVERY JOURNAL (BUFFER MEMORY IS {LOCAL | GLOBAL})

The RUJ buffers used by each process are normally allocated in local virtual memory. With the introduction of ROW CACHE, these buffers can now be assigned to a shared global section (GLOBAL memory) so that the recovery process can process this in memory buffer and possibly avoid a disk access.

This buffer memory can be defined a GLOBAL to improve ROW CACHE performance for recovery. If ROW CACHE is DISABLED then buffer memory is always LOCAL.

B.2.4.2 CACHE USING row-cache-name

Assigns the named row cache as the default physical row cache for all storage areas in the database. All rows stored in each storage area, whether they consist of table data, segmented string data, or special rows such as index nodes, are cached.

You must create the row cache before terminating the CREATE DATABASE statement. For example:

```
SQL> CREATE DATABASE FILENAME test_db
cont> ROW CACHE IS ENABLED
cont> CACHE USING test1
cont> CREATE CACHE test1
cont>    CACHE SIZE IS 100 ROWS
cont> CREATE STORAGE AREA area1;
```

If you do not specify the CACHE USING clause or the NO ROW CACHE clause, NO ROW CACHE is the default for the database.

You can override the database default row cache by either specifying the CACHE USING clause after the CREATE STORAGE AREA clause or by later altering the database and storage area to assign a new row cache. Only one physical area row cache is allowed for each storage area.

You can have multiple row caches containing rows for a single storage area by defining logical area row caches, where the row cache name matches the name of a table or index.

B.2.4.2.1 NO ROW CACHE

Specifies that the database default is not to assign a row cache to all storage areas in the database. You cannot specify the NO ROW CACHE clause if you specify the CACHE USING clause.

Alter the storage area and name a row cache to override the database default. Only one row cache is allowed for each storage area.

If you do not specify the CACHE USING clause or the NO ROW CACHE clause, NO ROW CACHE is the default for the database.

B.2.4.3 RESERVE n CACHE SLOTS

Specifies the number of row caches for which slots are reserved in the database.

You can use the RESERVE CACHE SLOTS clause to reserve slots in the database root file for future use by the ADD CACHE clause. Row caches can be added only if there are row cache slots available. Slots become available after a DROP CACHE clause or a RESERVE CACHE SLOTS clause.

The number of reserved slots for row caches cannot be decreased once the RESERVE clause is issued. If you reserve 10 slots and later reserve 5 slots, you have a total of 15 reserved slots for row caches.

Reserving row cache slots is an offline operation (requiring exclusive database access). See the Section B.1 for more information about row caches.

B.2.4.4 ROW CACHE IS ENABLED/ROW CACHE IS DISABLED

Specifies whether or not you want Oracle Rdb to enable the row caching feature.

When a database is created or is converted from a previous version of Oracle Rdb without specifying row cache support, the default is ROW CACHE IS DISABLED. Enabling row cache support does not affect database operations until a row cache area is created and assigned to one or more storage areas.

When the row caching feature is disabled, all previously created and assigned row caches remain in existence for future use when the row caching feature is enabled.

B.2.4.4.1 CHECKPOINT TIMED EVERY N SECONDS

Specifies the frequency with which the RCS process checkpoints the contents of the row caches back to disk. **The RCS process does not use the checkpoint frequency options of the FAST COMMIT clause.**

The frequency of RCS checkpointing is important in determining how much of an AIJ file must be read during a REDO operation following a node failure. It also affects the frequency that marked records get flushed back to the database, for those row caches that checkpoint to the database. The default is every 15 minutes (900 seconds).

B.2.4.4.2 CHECKPOINT ALL ROWS TO BACKING FILE/ CHECKPOINT UPDATED ROWS TO BACKING FILE/ CHECKPOINT UPDATED ROWS TO DATABASE

Specifies the default source and target for checkpoint operations for all row caches. If ALL ROWS is specified, then the source records written during each checkpoint operation are both the modified and the unmodified rows in a row cache. If UPDATED ROWS is specified, then just the modified rows in a row cache are checkpointed each time.

If the target of the checkpoint operation is BACKING FILE, then the RCS process writes the source row cache entries to the backing (.rdc) files. The row cache LOCATION, ALLOCATION, and EXTENT clauses are used to create the backing files. Upon recovery from a node failure, the database recovery process is able to re-populate the in-memory row caches from the rows found in the backing files.

If the target is DATABASE, then the target rows (only UPDATED ROWS is allowed) are written back to the database. The row cache LOCATION, ALLOCATION, and EXTENT clauses are ignored. Upon recovery from a node failure, the database recovery process has no data on the contents of the row cache. Therefore, it does not re-populate the in-memory row caches.

The CHECKPOINT clause of the CREATE CACHE, ADD CACHE, or ALTER CACHE clause overrides this database level CHECKPOINT clause.

B.2.4.4.3 LOCATION IS directory-spec Specifies the name of the default backing store directory to which all row cache backing files are written. The database system generates a file name automatically (row-cache-name.rdc) for each row cache backing file it creates when the RCS process first starts up. Specify a device name and directory name only, enclosed within single quotation marks. By default, the location is the directory of the database root file.

The LOCATION clause of the CREATE CACHE, ADD CACHE, or ALTER CACHE clause overrides this location which is the default for the database.

B.2.4.4.4 NO LOCATION Removes the location previously specified in a LOCATION IS clause for the database for the row cache backing file. If you specify NO LOCATION, the row cache backing file location becomes the directory of the database root file.

The LOCATION clause of the CREATE CACHE, ADD CACHE, or ALTER CACHE clause overrides this location which is the default for the database.

B.3 CREATE CACHE Clause

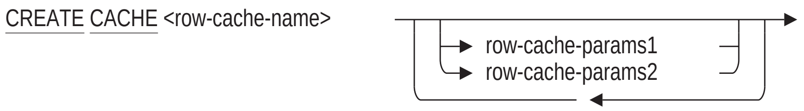
Creates a row cache area that allows frequently referenced rows to remain in memory even when the associated page has been transferred back to disk. This saves in memory usage because only the more recently referenced rows are cached versus caching the entire buffer.

See the Section B.1 and the Section B.2 for more information regarding the row cache areas.

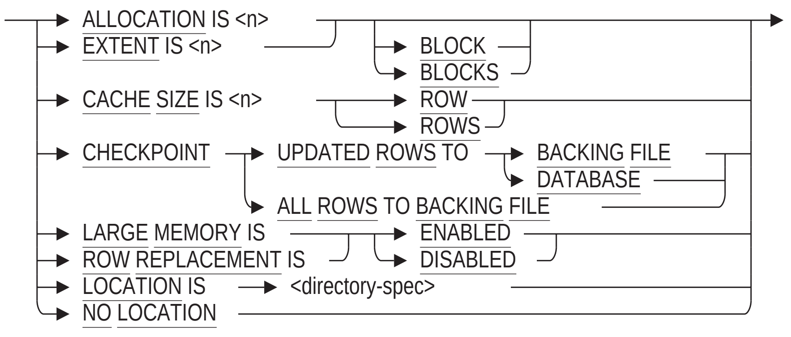
B.3.1 Environment

You can use the CREATE CACHE clause only within a CREATE DATABASE or IMPORT statement.

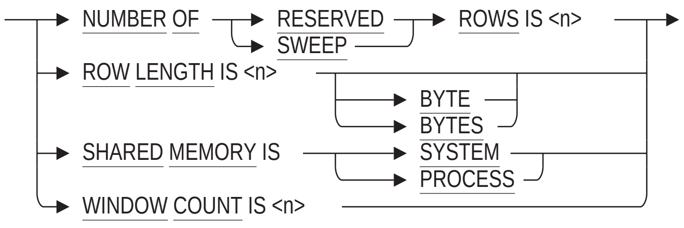
B.3.2 Format



row-cache-params1 =



row-cache-params2 =



B.3.3 Arguments

B.3.3.0.1 CACHE row-cache-name Creates a row cache.

B.3.3.0.2 ALLOCATION IS n BLOCK/ALLOCATION IS n BLOCKS Specifies the initial allocation of the row cache file (.rdc) to which cached rows are written during a checkpoint operation.

If the ALLOCATION clause is not specified, the default allocation in blocks is approximately 40 percent of the CACHE SIZE for this cache.

This clause is ignored if the row cache is defined to checkpoint to the database.

B.3.3.0.3 EXTENT IS n BLOCK/EXTENT IS n BLOCKS Specifies the file extent size for the row cache backing file (.rdc).

If the EXTENT clause is not specified, the default number of blocks is CACHE SIZE * 127 for this cache.

This clause is ignored if the row cache is defined to checkpoint to the database.

B.3.3.0.4 CACHE SIZE IS n ROW/CACHE SIZE IS n ROWS Specifies the number of rows allocated to the row cache. As the row cache fills, rows more recently referenced are retained in the row cache while those not referenced recently are discarded. Adjusting the allocation of the row cache helps to retain important rows in memory. If not specified, the default is 1000 rows.

The product of the CACHE SIZE and the ROW LENGTH settings determines the amount of memory required for the row cache. (Some additional overhead and rounding up to page boundaries is performed by the database system.) The row cache is shared by all processes attached to the database.

B.3.3.0.5 CHECKPOINT ALL ROWS TO BACKING FILE/ CHECKPOINT UPDATED ROWS TO BACKING FILE/ CHECKPOINT UPDATED ROWS TO DATABASE Specifies the source and target for checkpoint operations for the row cache. If ALL ROWS is specified, then the source records written during each checkpoint operation are both the modified and the unmodified rows in the row cache. If UPDATED ROWS is specified, then just the modified rows in the row cache are checkpointed each time.

If the target of the checkpoint operation is BACKING FILE, then the RCS process writes the source row cache entries to the backing (.rdc) files. The row cache LOCATION, ALLOCATION, and EXTENT clauses are used to create the backing files. Upon recovery from a node failure, the database recovery process is able to re-populate the in-memory row caches from the rows found in the backing files.

If the target is DATABASE, then the target rows (only UPDATED ROWS is allowed) are written back to the database. The row cache LOCATION, ALLOCATION, and EXTENT clauses are ignored. Upon recovery from a node failure, the database recovery process has no data on the contents of the row cache. Therefore, it does not re-populate the in-memory row caches.

This CHECKPOINT clause overrides the database level CHECKPOINT clause.

B.3.3.0.6 LARGE MEMORY IS ENABLED/LARGE MEMORY IS DISABLED Specifies whether or not large memory is used to manage the row cache. Very large memory (VLM) allows Oracle Rdb to use as much physical memory as is available. It provides access to a large amount of physical memory through small virtual address windows.

Use LARGE MEMORY IS ENABLED only when both of the following are true:

- You have enabled row caching.
- You want to cache large amounts of data, but the cache does not fit in the virtual address space.

The default is DISABLED.

See the Usage Notes for restrictions pertaining to the very large memory (VLM) feature.

B.3.3.0.7 ROW REPLACEMENT IS ENABLED/ROW REPLACEMENT IS DISABLED Specifies whether or not Oracle Rdb replaces rows in the cache. When the ROW REPLACEMENT IS ENABLED clause is used, rows are replaced when the row cache becomes full. When the ROW REPLACEMENT IS DISABLED clause is used, rows are not replaced when the cache is full. The type of row replacement policy depends upon the application requirements for each cache.

The default is ENABLED.

B.3.3.0.8 LOCATION IS directory-spec Specifies the name of the default backing store directory to which all row cache backing files are written. The database system generates a file name automatically (row-cache-name.rdc) for each row cache backing file it creates when the RCS process first starts up. Specify a device name and directory name only, enclosed within single quotation marks. By default, the location is the directory of the database root file.

This LOCATION clause overrides a previously specified location at the database level.

This clause is ignored if the row cache is defined to checkpoint to the database.

B.3.3.0.9 NO LOCATION Removes the location previously specified in a LOCATION IS clause for the database for the row cache backing file. If you specify NO LOCATION, the row cache backing file location becomes the directory of the database root file.

This clause is ignored if the row cache is defined to checkpoint to the database.

B.3.3.0.10 NUMBER OF RESERVED ROWS IS n Specifies the maximum number of cache rows that each user can reserve. The default is 20 rows.

The number of reserved rows parameter is also used when searching for available slots in a row cache. The entire row cache is not searched on the initial pass. This parameter is used as the maximum number of rows that are searched for a free slot. If at least one free slot is found, the insert operation can proceed. If no free slots are found in this initial search, Oracle Rdb will continue searching through the cache until it finds a free slot.

B.3.3.0.11 NUMBER OF SWEEP ROWS IS n Specifies the number of modified cache rows that will be written back to the database to make space available in the cache for subsequent transactions to insert rows into the cache. It is recommended that users initially specify the number of sweep rows to be between ten and thirty percent of the total number of rows in the cache. Users should then monitor performance and adjust the number of sweep rows if necessary. The default setting is 3000 rows.

B.3.3.0.12 ROW LENGTH IS n BYTE/ROW LENGTH IS n BYTES Specifies the size of each row allocated to the row cache. Rows are not cached if they are longer than a row cache row. The ROW LENGTH is an aligned longword rounded up to the next multiple of 4 bytes.

If the ROW LENGTH clause is not specified, the default row length is 256 bytes. The maximum row length in a row cache area is 65535 bytes.

If the ROW LENGTH clause is not specified, the default row length is 256 bytes.

B.3.3.0.13 SHARED MEMORY IS SYSTEM/SHARED MEMORY IS PROCESS Determines whether cache global sections are created in system space or process space. The default is SHARED MEMORY IS PROCESS.

When you use cache global sections created in the process space, you and other users share physical memory and the OpenVMS Alpha operating system maps a row cache to a private address space for each user. As a result, all users are limited by the free virtual address range and each use a percentage of memory in overhead. If many users are accessing the database, the overhead can be high.

When many users are accessing the database, consider using SHARED MEMORY IS SYSTEM. This gives users more physical memory because they share the system space of memory and there is none of the overhead associated with the process space of memory.

B.3.3.0.14 WINDOW COUNT IS n Specifies the number of virtual address windows used by the LARGE MEMORY clause.

The window is a view into the physical memory used to create the very large memory (VLM) information. Because the VLM size may be larger than that which can be addressed by a 32-bit pointer, you need to view the VLM information through small virtual address windows.

You can specify a positive integer in the range from 10 through 65535. The default is 100 windows.

B.3.4 Usage Notes

- If the name of the row cache is the same as any logical area (for example a table name, index name, storage map name, RDB$SEGMENTED_STRINGS, RDB$SYSTEM_RECORD, and so forth), then this is a logical area cache and the named logical area is cached automatically. Otherwise, a storage area needs to be associated with the cache.
- The CREATE CACHE clause does not assign the row cache to a storage area. You must use the CACHE USING clause with the CREATE STORAGE AREA clause of the CREATE DATABASE statement or the CACHE USING clause with the ADD STORAGE AREA or ALTER STORAGE AREA clauses of the ALTER DATABASE statement.
- The product of the CACHE SIZE and the ROW LENGTH settings determines the amount of memory required for the row cache (some additional overhead and rounding up to page boundaries is performed by the database system).
- The row cache is shared by all processes attached to the database on any one node.
- The following are requirements when using the row caching feature:
  - After-image journaling must be enabled
  - Fast commit must be enabled
  - Number of cluster nodes must equal 1
- Use the SHOW CACHE statement to view information about a cache.

C
Release Notes Relating to the Row Cache Feature

This section describes software errors that were fixed by Oracle Rdb7 Release 7.0.1.5 and 7.0.1.6 relating specifically to the row cache feature.

C.1 Software Errors Fixed That Apply to All Interfaces

C.1.1 RCS Maximum Log File Size Control Logical

In prior versions of Oracle Rdb7, the Row Cache Server (RCS) process log file (enabled via the RDM$BIND_RCS_LOG_FILE logical name) would continue to grow until the database was shut down. This would be a significant problem because when the disk containing the log file would become full, the RCS process could fail.

The RCS process log file maximum size can now be controlled with the system logical name RDM$BIND_RCS_LOG_REOPEN_SIZE. This logical, when defined before the database is opened, limits the allocated size of the RCS log file. When the log file allocation reaches the specified number of disk blocks, the current log file will be closed and a new log file opened. Older log files can be archived or purged as needed.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.5.

C.1.2 New RMU /SET ROW_CACHE [/ENABLE | /DISABLE] Command

A new RMU /SET command “ROW_CACHE” has been added to allow the database Row Cache feature to be enabled or disabled without requiring that the database be opened. This command requires exclusive database access (the database can not be open or be accessed by other users).

Valid qualifiers for the “RMU /SET ROW_CACHE” command are:

- /ENABLE to enable row caching
- /DISABLE to disable row caching
- /LOG to display a log message at the completion of the RMU /SET operation

The /ENABLE and /DISABLE qualifiers are mutually exclusive.

This command has been added to Oracle Rdb7 Release 7.0.1.5.

C.1.3 RCS Clearing "GRIC" Reference Counts

When the Oracle Rdb7 Row Cache feature is enabled, the Row Cache Server (RCS) process will attempt to clear the reference count field in a data structure called a GRIC. The reference count will be cleared periodically based on the number of DBR (Database Recovery) processes run. If enough DBR processes have run, a Row Cache "sweep" request can trigger the reference count clearing.

When a process that uses a row cache abnormally terminates (via STOP/ID, for example), it can leave references in the cache that would prevent rows in the cache from being removed. This can cause the cache to become full of rows that are not really referenced by any process though they appear to be referenced due to an elevated reference count.

A Row Cache "sweep" request to the RCS process indicates that a cache is "full" and there is no more room to insert new rows into the cache. When the RCS process receives the sweep request, it will see if a number of DBRs have run since the last sweep. If enough DBRs have run (the default is 25 DBRs since the last sweep for the cache), the RCS will initiate a "Release GRICs" operation.

This operation can have a minor performance impact to users of the cache and can also delay the RCS from performing other operations. This is why it is a periodic event.

The system logical name RDM$BIND_RCS_CLEAR_GRICS_DBR_CNT can be used to control the number of DBRs that must elapse before the RCS will initiate clearing of the GRIC reference counts. The maximum value of the logical name is "100000". The default value (if the logical name is not defined) is "25". Defining the logical name with a value of "0" disables clearing the reference counts.

For most systems, the default value is adequate. However, systems with very frequent database recoveries may need a high value of the logical name to reduce the frequency that the reference counts are cleared. The RCS process log file can be used to determine how often the reference counts are cleared.

This new logical name has been included in Oracle Rdb7 Release 7.0.1.5.

C.1.4 Row Cache RDC File Name Change

In the previous release of Oracle Rdb7, the Row Cache backing store file used a file type of “.RDC”. This behavior caused a file name conflict when a database was replicated either with the RMU/COPY command or when using the “Hot Standby” feature.

This conflict has been resolved in Oracle Rdb7 Release 7.0.1.5. The Row Cache backing store file type has been extended to include the root file device name and file ID in a BASE32 format (where valid characters are 0 to 9 and A to W).

For example, a row cache backing store file name may now have a format similar to the following:

```
EMPIDX_10_0.RDC_0C1H85848NO00063228L;1
```

In this example, the value “0C1H85848NO00063228L” represents the device name and file ID of the root file for the database. The file type is always prefixed with “.RDC_” All Row Cache backing store files for a database have this same exact file type. Another database using the same location for backing store files would use a different file type (perhaps “.RDC_4D87HD234FSD0063228L”).

To associate a database with a Row Cache backing store file, the “RMU /DUMP /CACHE_FILE” command can be used to display the Row Cache backing store file header when the full name of the database root file is stored.

Because existing Row Cache backing store files have a file type of “.RDC”, if you use the RDM$BIND_RCS_KEEP_BACKING_FILES logical to keep existing backing store files from being deleted when a database is closed, you should deassign the logical prior to closing the database(s) in preparation for installing Oracle Rdb7 Release 7.0.1.5. This will allow existing “.RDC” files to be deleted properly.

C.1.5 VLM or System Space Buffer Corruption

Very rarely, small portions of cache memory could be incorrectly left un-initialized when using the Row Cache Feature with the Very Large Memory (VLM) or System Space Buffers (SSB) options on multi-processor (SMP) Alpha systems.

This problem could occur more often with large caches, under heavy system loads, on multi-processor systems. If the process that was initializing a row cache was rescheduled onto another CPU, it was possible that the CPU translation buffer (TB) on one of the processors was not correctly invalidated. If the process were to be rescheduled back on to the original processor, there was an outside chance that a memory page within the cache would not be correctly erased.

Because the first process to access a row cache creates and initializes the cache, a possible workaround is to stop all but the primary CPU on the system while row caches are being initially accessed.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.6. During VLM or SSB creation, the process prevents itself from being rescheduled while it is invalidating the system translation buffers.

C.1.6 Invisible Row After Erase and Store With Row Cache

If a row cache was created with a row length smaller than the largest data row to be stored and if the row had previously been erased from the cache, it was possible for the row to become “invisible”.

The following example demonstrates the problem.

```
SQL> create data file foo
cont> number of cluster nodes is 1
cont> reserve 1 cache slot;
SQL> create table c1 (t1 char (100));
SQL> commit;
SQL> disconnect all;
SQL> alter data file foo
cont> row cache enable
cont> add journal j1 file j1
cont> journal enable (fast commit enable)
cont> add cache c1 row length is 50;
SQL> attach 'file foo';
SQL> insert into c1
cont> values ('ab')
cont> returning dbkey;
                  DBKEY
               47:554:0
1 row inserted
SQL> commit;
SQL> delete from c1;
1 row deleted
SQL> commit;
SQL> disconnect all;
SQL> attach 'file foo';
SQL> insert into c1
cont> values ('abababababababababababaababababababababababa')
cont> returning dbkey;
                  DBKEY
               47:554:0
1 row inserted
SQL> commit;
SQL> select * from c1;
0 rows selected
SQL> commit;
```

This problem has been corrected in Oracle Rdb7 Release 7.0.1.6. The erased row is now correctly detected and the row that is too large for the cache is now returned from disk.

C.1.7 Overriding RCS Checkpoint Timer Interval

In the prior versions of Oracle Rdb7, the Row Cache Server (RCS) process' checkpoint timer interval could be overridden by the system logical "RDM$BIND_CKPT_TIME". This is the logical that allows the fast commit checkpoint timer interval to be overridden. Using the same logical for the RCS checkpoint timer was confusing and error prone.

Beginning with Oracle Rdb7 Release 7.0.1.6, the RCS process' checkpoint timer interval can be overridden with a new system logical name, "RDM$BIND_RCS_CKPT_TIME".

If neither this logical nor the "ROW CACHE IS ENABLED (CHECKPOINT TIMED EVERY n SECONDS)" database clause is specified, then the RCS process will use the "RDM$BIND_CKPT_TIME" logical name or its associated dashboard value.

If RCS still has a zero checkpoint timer interval, then it will default to a fixed 15 minute interval.

C.1.8 Refresh RCS Metadata Information

In the prior versions of Oracle Rdb7, the Row Cache Server (RCS) process would maintain its metadata structures across checkpoint and sweep requests. While the RCS process was active, however, Oracle Rdb7 would allow tables, indices, and storage areas to be dropped and recreated. In these situations, it was possible for the RCS process to not notice the metadata changes and use the original metadata to write modified records from the row caches back to the original database storage areas. This would result in database corruptions and bugcheck dumps.

In Oracle Rdb7 Release 7.0.1.6, the RCS now recognizes that if it is not holding the corresponding logical or physical area locks, its metadata may be obsolete. When this occurs, the RCS process refreshes its metadata structures from the AIP and root file information.

C.1.9 RCS ACCVIO When Checkpointing All Row Caches to Database

Begining with Release 7.0.1.5 of Oracle Rdb7, the Row Cache Server (RCS) process would inadvertently access violate after completing its final checkpoint to the database as part of a database shutdown operation. It was access violating while trying to clean up a data structure that had not been allocated.

This problem does not corrupt the database. Simply reopen the database and database access will be fine until the database is closed again, whereupon this problem will be hit again. A workaround to this problem is to have at least one row cache checkpoint to a backing file.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.6.

D
Known Problems and Restrictions Relating to the Row Cache Feature

This section describes known problems and restrictions relating to the row cache feature and includes workarounds where appropriate. Unless otherwise noted, all notes apply to all platforms.

D.1 Known Problems and Restrictions

D.1.1 RMU Online Verification Operations and Row Cache

When using row caches, some RMU online verification operations may report errors in the database structure and may not be generally reliable in all verifications. These errors may be due to RMU validating the on-disk database structure and not the actual logical database structure including the row cache contents.

For example, one of the verifications that is performed by RMU/VERIFY is to ensure that system records in mixed format areas have a “system record” record ID. However, when a physical row cache is being used, the row on the database page may be marked as “reserved by record cache” because the row has been modified in the row cache but has not yet been flushed to disk.

In the following example, the database ID of **00002011** refers to the “reserved by record cache” record type and **00002001** refers to the system record type:

```
$ RMU/VERIFY/ONLINE DKA0:[DB]MYDB.RDB;1
%RMU-E-PAGSYSREC, area INDEX_MIXED_AREA, page 3
                  system record contains an invalid database ID
                  expected: 00002001 (hex), found: 00002011 (hex)
```

D.1.2 Limitation: Online RMU /VERIFY and Row Cache

Performing online RMU /VERIFY operations on a database with the Row Cache feature enabled may report errors even though there is actually no problem. RMU /VERIFY is not fully integrated with the Row Cache feature in this release. Because of this, if there is database modification activity occurring while the verify is running, misleading error messages may be displayed.

If possible, limit online RMU /VERIFY operations to times when the database is not being actively modified or perform an offline database verification.

This problem will be corrected in a future Oracle Rdb release.

D.1.3 Adding Row Caches Requires Exclusive Database Access

Adding a row cache with the ALTER DATABASE ADD CACHE command now requires exclusive database access.

Previously, it was possible for a new row cache to be added online. This new cache would be seen by users attaching to the database after the cache was created, but users that were already attached to the database would not be able to access the cache and would return results from the database without referencing the cache. This situation resulted in database corruption.

D.1.4 Conflicts When Caching Metadata and Executing Certain SQL Database Operations

When caching metadata, you will experience conflicts when executing database operations through SQL that require exclusive database access. For example, adding new row caches or dropping existing ones requires exclusive database access. When the SQL command is parsed, the Oracle Rdb system tables are queried. This access to the system tables creates the row caches and causes the RCS process to come up to manage those row caches. As a result, the database now has another “user”, the RCS process. This causes the exclusive database operation to fail.

To resolve this, you must first turn off row caching temporarily using the RMU Set command specifying the Row_Cache and Disabled qualifiers. Then, perform the SQL operation that requires exclusive database access. Finally, re-enable row caching using the RMU Set command with the Row_Cache and Enabled qualifiers.

E
Logical Names Relating to the Row Cache Feature

This section describes logical names relating specifically to the row cache feature and explains when and how to use them. Note that the fields following the logical name list the table name in which the logical must be defined and the value of the logical with defaults given where applicable.

E.1 RDM$BIND_CKPT_FILE_SIZE

```
RDM$BIND_CKPT_FILE_SIZE          LNM$FILE_DEV             INTEGER
```

This logical represents the percentage of the row cache size that you want the backing file allocation to be. Applied to all backing files. This overrides the backing file's allocation specified in the CREATE/ADD CACHE definition.

E.2 RDM$BIND_CKPT_TIME

```
RDM$BIND_CKPT_TIME               LNM$FILE_DEV             INTEGER  (Default=0)
```

This logical represents the frequency of RCS checkpoint. It overrides the "Alter database row cache is enabled (checkpoint timed every N seconds)" value.

E.3 RDM$BIND_DBR_UPDATE_RCACHE

```
RDM$BIND_DBR_UPDATE_RCACHE       LNM$SYSTEM_TABLE          0 or 1(Default)
```

If the logical is set to 0, during recovery from node failure, don't repopulate in-memory row caches from their backing files (only recover the database). If the logical is set to 1 (the default), during recovery from node failure, repopulate in-memory row caches from backing files and from REDO operations.

E.4 RDM$BIND_RCACHE_INSERT_ENABLED

```
RDM$BIND_RCACHE_INSERT_ENABLED  LNM$FILE_DEV              0 or 1(Default)
```

This is a process logical. If the logical is set to 0, this process cannot insert any rows into the row caches; this process can only use what is already there. If the logical is set to 1 (the default), the process can insert new rows into the row cache, if they fit.

E.5 RDM$BIND_RCACHE_LATCH_SPIN_COUNT

```
RDM$BIND_RCACHE_LATCH_SPIN_COUNT   LNM$FILE_DEV           INTEGER  (Default=1024)
```

This logical represents how many iterations to retry getting the row cache latch before hibernating. This consumes CPU but can acquire the latch faster. Set in 1000s.

E.6 RDM$BIND_RCACHE_RCRL_COUNT

```
RDM$BIND_RCACHE_RCRL_COUNT      LNM$FILE_DEV            INTEGER  (Default=0)
```

This logical represents the number of rows to reserve when acquiring empty slots in a row cache. This overrides the “NUMBER OF RESERVE ROWS IS N” clause.

E.7 RDM$BIND_RCS_BATCH_COUNT

```
RDM$BIND_RCS_BATCH_COUNT        LNM$SYSTEM_TABLE        INTEGER  (Default=3000)
```

This logical represents the number of rows RCS attempts to write out at a time during the course of a checkpoint or sweep.

E.8 RDM$BIND_RCS_CARRYOVER_ENABLED

```
RDM$BIND_RCS_CARRYOVER_ENABLED  LNM$SYSTEM_TABLE        0 or 1(Default)
```

If the logical is set to 0, RCS doesn't honor carryover locks for logical/physical areas. It continues to hold them (good for RCS performance, but prevents exclusive access to these logical/physical areas). If the logical is set to 1 (the default), RCS honors carryover locks and gives up logical/physical area locks it is holding that it is not using but that simply remain from a prior operation.

E.9 RDM$BIND_RCS_CKPT_COLD_ONLY

```
RDM$BIND_RCS_CKPT_COLD_ONLY     LNM$SYSTEM_TABLE        0(Default) or 1
```

If the logical is set to 0 (the default), checkpoint/sweep all marked records in a row cache. If the logical is set to 1, only checkpoint records marked before the PRIOR ckpt interval (only checkpoint the older/colder data, but this also keeps the RCS ckpt farther behind causing more AIJ to read during REDO).

E.10 RDM$BIND_RCS_CKPT_BUFFER_CNT

```
RDM$BIND_RCS_CKPT_BUFFER_CNT    LNM$SYSTEM_TABLE        INTEGER (Default=15)
```

This logical represents the number of buffers to use to write records to backing files during checkpoints.

E.11 RDM$BIND_RCS_CKPT_TIME

```
RDM$BIND_RCS_CKPT_TIME           LNM$SYSTEM_TABLE       INTEGER (Default=0)
```

This logical overrides the RCS process' checkpoint timer interval. This logical was added in Release 7.0.1.6. If neither this logical nor the “ROW CACHE IS ENABLED (CHECKPOINT TIMED EVERY n SECONDS)” database clause is specified, then the RCS process will use the “RDM$BIND_CKPT_TIME” logical name or its associated dashboard value. If RCS still has a zero checkpoint timer interval, then it will default to a fixed 15 minute interval.

E.12 RDM$BIND_RCS_CLEAR_GRICS_DBR_CNT

```
RDM$BIND_RCS_CLEAR_GRICS_DBR_CNT  LNM$SYSTEM_TABLE      INTEGER (Default=25)
```

This logical represents the frequency (based on the number of DBR processes that run) with which the RCS will attempt to release references in the cache left by abnormally terminated processes. For each sweep request for a cache, if at least this number of DBR processes have run since the last sweep for the cache, the RCS will initiate a "Release GRICs" operation. This operation can have a minor performance impact to users of the cache and can also delay the RCS from performing other operations. This is why it is a periodic event. The maximum value of the logical is 100000. The default value is 25. Defining the logical name with a value of 0 will disable the clearing of reference counts.

E.13 RDM$BIND_RCS_CREATION_IMMEDIATE

```
RDM$BIND_RCS_CREATION_IMMEDIATE   LNM$SYSTEM_TABLE        0(Default) or 1
```

If the logical is set to 0 (the default), for automatic open database, create RCS process on first reference to a row cache. If the logical is set to 1, for automatic open database, create RCS process on initial attach. If the logical is set to 1, for manual open database, RCS is started immediately.

E.14 RDM$BIND_RCS_KEEP_BACKING_FILES

```
RDM$BIND_RCS_KEEP_BACKING_FILES   LNM$SYSTEM_TABLE        0(Default) or 1
```

If the logical is set to 0 (the default), the RCS creates/deletes backing files on each startup/shutdown. If the logical is set to 1, the RCS retains backing files on shutdown and reuses them on startup.

E.15 RDM$BIND_RCS_LOG_FILE

```
RDM$BIND_RCS_LOG_FILE           LNM$SYSTEM_TABLE          File Name
```

This logical specifies the location and name of the optional RCS process log file. If the logical is not defined, no RCS logging is done. It is recommended that logging be turned on. If a location is not specified along with the file name, the log file is created in the same location as the database root file.

E.16 RDM$BIND_RCS_LOG_HEADER

```
RDM$BIND_RCS_LOG_HEADER     LNM$SYSTEM_TABLE              0 or 1(Default)
```

If the logical is set to 0, don't insert header sections in RCS log file. If the logical is set to 1 (the default), insert normal header sections into the RCS log file.

E.17 RDM$BIND_RCS_LOG_REOPEN_SIZE

```
RDM$BIND_RCS_LOG_REOPEN_SIZE    LNM$SYSTEM_TABLE         INTEGER  (Default=0)
```

This logical represents the maximum block size of the RCS log file before the RCS opens a new log file.

E.18 RDM$BIND_RCS_LOG_REOPEN_SECS

```
RDM$BIND_RCS_LOG_REOPEN_SECS    LNM$SYSTEM_TABLE         INTEGER  (Default=0)
```

This logical, when defined before the database is opened, causes the RCS log file to be reopened after every 'n' seconds as specified by the value of the logical name. If the value of the logical is 0 or it is not defined, then the RCS Log file is not reopened based on time. The maximum value allowed is 31449600 (which is one year noted in seconds).

E.19 RDM$BIND_RCS_PRIORITY

```
RDM$BIND_RCS_PRIORITY           LNM$SYSTEM_TABLE         INTEGER
```

This logical represents the base priority of the RCS process.

E.20 RDM$BIND_RCS_SWEEP_COUNT

```
RDM$BIND_RCS_SWEEP_COUNT         LNM$SYSTEM_TABLE         INTEGER
```

This logical represents the number of rows to sweep. It overrides the "NUMBER OF SWEEP ROWS IS N" clause.

E.21 RDM$BIND_RCS_VALIDATE_SECS

```
RDM$BIND_RCS_VALIDATE_SECS       LNM$SYSTEM_TABLE         INTEGER
```

This logical defines the number of seconds between each cache validation pass. A value in the range of 300 (5 minutes) to 86400 (24 hours) is suggested. A value of 0 disables the cache validations. Once initiated, the interval can be re-set by changing the logical name definition; the logical is translated at each validation.

E.22 RDM$BIND_RUJ_GLOBAL_SECTION_ENABLED

```
RDM$BIND_RUJ_GLOBAL_SECTION_ENABLED   LNM$SYSTEM_TABLE    0 or 1
       (Default=1 if row cache enabled)
       (Default=0 if row cache disabled)
```

If the logical is set to 0, don't place RUJ I/O buffers in global section so DBR can see them. If the logical is set to 1, place RUJ I/O buffers in global section so DBR can see them.