Oracle Rdb7™ for OpenVMS

Release Notes

Release 7.0.3

August 1999

ORACLE®

Oracle Rdb7 Release Notes

Release 7.0.3

This document was prepared using VAX DOCUMENT Version 2.1.

Contents

Tables

Preface

Purpose of This Manual

This manual contains release notes for Oracle Rdb7 Release 7.0.3. 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 7.0.3.

Document Structure

This manual consists of five chapters:

Chapter 1	Describes how to install Oracle Rdb7 Release 7.0.3.
Chapter 2	Describes software errors corrected in Oracle Rdb7 Release 7.0.3.
Chapter 3	Describes software errors corrected in Oracle Rdb7 Release 7.0.2.1.
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 7.0.3.

1 Installing Oracle Rdb7 Release 7.0.3

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 7.0.3 are:

- RDBSC070 for Oracle Rdb7 for OpenVMS VAX standard version.
- RDBASC070 for Oracle Rdb7 for OpenVMS Alpha standard version.
- RDBMVC070 for Oracle Rdb7 for OpenVMS VAX multiversion.
- RDBAMVC070 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:[RDBSC070.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 RDBSC070 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, “RDBVMSV7.0-3”.

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=RDBVMSV7.0-3 -
   _$ /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 New Documentation HTML Save Sets Available

Included with this release is a new backup save set (RDB_702_HTML.BCK) and a new self-extracting archive file (RDB_702_HTML.EXE) for Windows NT and Windows 95. These new files contain the Oracle Rdb V7.0 (and related products) documentation in HTML format. Documentation is included for the following products:

- Oracle Rdb, Release 7.0
- Rdb Web Agent, Release 2.2
- SQL*Net for Rdb7, Release 7.1.2
- Hot Standby for Oracle Rdb and CODASYL DBMS, Release 7.0

- Distributed Option for Rdb, Release 7.0

When you expand the RDB_702_HTML.BCK backup save set, the WWW and DOC sub-directories are created and product-specific sub-directories are created below DOC. Be sure to maintain the directory structure by specifying the following command:

```
$ BACKUP RDB_702_HTML.BCK/SAVE disk:[directory...]
```

To access this library of documentation, point to LIBRARY.HTML using your favorite web browser.

When you expand the RDB_702_HTML.EXE self-extracting archive file for your Windows NT or Windows 95 system, the rdbhtmldocs directory is created with product-specific directories below that. Again, access this library of documentation by pointing to LIBRARY.HTML from your favorite web browser.

The PostScript format of this documentation is available in the RDB7PS.BCK backup save set.

2 Software Errors Fixed in Oracle Rdb7 Release 7.0.3

This chapter describes software errors that are fixed by Oracle Rdb7 Release 7.0.3.

2.1 Software Errors Fixed That Apply to All Interfaces

2.1.1 Alpha EV6 Processor Support Added

As of this release of Rdb7, 7.0.3, the Alpha EV6 processor is supported. This is the only thing that has changed between Oracle Rdb7 Release 7.0.2.1 and Oracle Rdb7 Release 7.0.3.

2.1.2 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. Rdb has always had a minimum VMS version requirement. With 7.0.1.5 and for all future Rdb releases, we have expanded this concept to include a maximum VMS version check. and a maximum suppported 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 this release of Rdb7, 7.0.3, the Alpha EV6 processor is supported.

The check for the OpenVMS operating system version and supported hardware platforms is performed both at installation time and at run-time. 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 run-time, Oracle Rdb will not start.

3 Software Errors Fixed in Oracle Rdb7 Release 7.0.2.1

This chapter describes software errors that are fixed by Oracle Rdb7 Release 7.0.2.1.

3.1 Software Errors Fixed That Apply to All Interfaces

3.1.1 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. Rdb has always had a minimum VMS version requirement. With 7.0.1.5 and for all future Rdb releases, we have expanded this concept to include a maximum VMS version check. The reason for this check is to improve product quality.

OpenVMS Version 7.2-n is the maximum supported version of OpenVMS and Alpha EV56 is the maximum supported processor for Oracle Rdb7 Release 7.0.2.1.

The new Alpha EV6 processor was introduced in OpenVMS Version 7.1-2. While OpenVMS Version 7.1-2 is supported by Oracle Rdb, the Alpha EV6 processor is not. Oracle Rdb has not been certified on the new Alpha EV6 processor. Our goal is to certify new versions of OpenVMS and new processors, such as EV6, as soon as sufficient hardware is made available to us by Compaq Computer Corporation.

The check for the OpenVMS operating system version and supported hardware platforms is performed both at installation time and at run-time. 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 run-time, Oracle Rdb will not start.

3.1.2 Query With IN Clause Bugchecks

Bug 856818

The following query with IN clause bugchecks.

```
Select
        T3_IMP_CRE , T3_IMP_RES , T1_COSTO_RICARICA
from    T1, T2, T3
where
        T3_PRRTG_COD = T1_PRRTG_COD     ! transitive betw T3 & T1
 and    T3_TORTG_COD = T2_COD           ! transitive betw T3 & T2
        ! Selection predicates on T3
 and    T3_DATA_OPE between T1_DATA_INI and T1_DATA_FIN
 and    T3_DATA_OPE between '01-Nov-1997 00:00:00.00'
                         And    '11-Nov-1997 23:59:59.00'
 and    T3_TORTG_COD in ('002','005','006','007')
 and    T3_DEAL_COD = 5883
;
```

A workaround to this problem is to use the set SQL command SET FLAGS 'NOTRANSITIVITY' to turn off transitivity.

This problem has been corrected in Oracle Rdb7 Release 7.0.2.1.

3.1.3 Query With OR & AND Constant Predicates Gets Zero Costing

Bug 876813

The following query with constant predicates is not properly optimized in the costing when the SQL set flags 'COSTING' is turned on.

```
select * from departments where
  (department_code='ADMN' and 8<>9) or
  (department_code='ADMN' and 8=9);

...etc..
--- Start tracing CRTV cost for context: 0

Index name: "DEPARTMENTS_INDEX" Context:0 Table:"DEPARTMENTS"

Index Retrieval Seg 0, [1:1] "Direct Lookup"    TYPE"=Reg"      Unique Ndx
Equality        Column: "DEPARTMENT_CODE" (context 0)
CRTV$V_NEED_BITVEC - @ X00A71F78: 00000000 00000000 00000000 00000000
Retrievals per CRTV predicate for all scans/lookups  1.0000000E+00
Startup Cost for sorted index  1.0000000E+00
Calculating cost to descend B-tree...
Cost after descending to level_1 node  1.4242887E+00
Calculating cost to fetch data records...
Retrievals per all indexed predicates  0.0000000E+00
Retrieval cost using KEY_GROUP_FACTOR & DATA_CLUST_FACTOR  0.0000000E+00
Final Cost after data fetch  1.4242887E+00
--- CRTV list end

DEPARTMENT_CODE  DEPARTMENT_NAME          MANAGER_ID BUDGET_PROJECTED   BUDGET_ACTUAL
ADMN             Corporate Administration  00225          NULL            NULL

1 row selected
```

There is no workaround for this problem.

This problem has been corrected in Oracle Rdb7 Release 7.0.2.1.

3.1.4 Storage Area List Sometimes Truncated by SQL and RMU/EXTRACT

Bug 633334

In previous versions of Oracle Rdb, the output from the SQL SHOW STORAGE AREA or the RMU/EXTRACT storage area output could be truncated if the database had many storage areas. This problem also affected the exported storage areas so that IMPORT might fail to create a complete database.

For instance, one user with 2132 storage areas in use found that only 238 were displayed by SHOW STORAGE AREA. SQL and RMU use a special interface call to retrieve all the names of the storage areas. This problem occurs when the buffer is incorrectly truncated. The actual number of storage areas displayed depends upon the length of the names and will vary from database to database.

This problem was corrected in Oracle Rdb7 Release 7.0.1.6 but the release note was inadvertently left out. The buffer returned from the storage area query is no longer truncated.

3.1.5 GROUP BY Query With SUM Aggregate Function Returns Wrong Result

Bug 874047

The following GROUP BY query with SUM aggregate function returns the wrong result for the unit_delivery_month column.

```
select
       d.unit_no,
       u.unit_delivery_month,
       sum(u.unit_combined_cost)
  from
    units_details d, units u
  where
    d.unit_no = u.unit_no
  group by
       d.unit_no,
       u.unit_delivery_month;
   D.UNIT_NO   U.UNIT_DELIVERY_MONTH
      301071   1991-12                                20
      304823   1991-12                                40
2 rows selected
```

```
Where the following is the script to reproduce:
```

```
create table units (
        unit_prefix char(4),
        unit_no integer,
        unit_cost integer,
        unit_delivery_date date ansi,
        unit_delivery_month computed by
           cast(UNIT_DELIVERY_DATE AS CHAR(7)),
        unit_combined_cost computed by
          (unit_cost * 2)
                   );

insert into units (unit_prefix, unit_no, unit_cost,
                   unit_delivery_date)
  values ('SCZU',301071, 10, DATE'1990-12-07');
insert into units (unit_prefix,unit_no, unit_cost,
                   unit_delivery_date)
  values ('SCZU',304823, 20, DATE'1991-12-13');

create table units_details (unit_no integer, unit_prefix char(4),
                            unit_type char(5));
insert into units_details (unit_no,unit_prefix,unit_type)
  values (304823,'SCZU','BX4');
insert into units_details (unit_no,unit_prefix,unit_type)
  values (301071,'SCZU','BX4');
commit;
```

There is no workaround for this problem.

This problem has been corrected in Oracle Rdb7 Release 7.0.2.1.

3.1.6 Enhancements to ALTER STORAGE MAP and ALTER INDEX

Bug 882000

When a table's storage map has the attribute PARTITIONING IS NOT UPDATABLE, then mapping of data to a storage area is strictly enforced. This is known as **strict partitioning**. This release of Oracle Rdb7 lifts restrictions imposed by earlier releases.

These enhancements allow more flexible ALTER STORAGE MAP and ALTER INDEX usage, often with reduced I/O requirements.

- ALTER STORAGE MAP now allows a map to specify PARTITIONING IS NOT UPDATABLE to change a storage map to be strictly partitioned. Previous versions reported an error as shown in the following example:

  ```
  SQL> -- make this a strictly partitioned map
  SQL> alter storage map S1
  cont>     partitioning is not updatable
  cont>     store using (C1)
  cont>         in JOBS with limit of (100)
  cont>         in DEPARTMENTS with limit of (200)
  cont>         in EMP_INFO with limit of (300);
  %RDB-E-NO_META_UPDATE, metadata update failed
  -RDMS-E-WISH_LIST, feature not implemented yet
  ```

  This operation now succeeds. However, it will require a full scan of the table since Rdb does not know if the rows still conform to the original partitioning scheme.

- ALTER STORAGE MAP now supports adding areas to a strictly partitioned map without requiring a rebuild of the table if they are appended to the end of the old map. The following example shows that no database I/O is performed to restructure the table.

  ```
  SQL> set flags 'stomap_stats';
  SQL> create table T1 (C1 integer);
  SQL> create storage map S1
  cont>     for T1
  cont>     partitioning is not updatable
  cont>     store using (C1)
  cont>         in JOBS with limit of (100)
  cont>         in DEPARTMENTS with limit of (200)
  cont>         in EMP_INFO with limit of (300);
  ~As: create storage map "S1"
  ~As: Table "T1" (sys=0, rest=0, tmptbl=0)
     .
     .
     .
  SQL> -- add new area to the end of the map
  SQL> -- no rebuild required
  SQL> alter storage map S1
  cont>     partitioning is not updatable
  cont>     store using (C1)
  cont>         in JOBS with limit of (100)
  cont>         in DEPARTMENTS with limit of (200)
  cont>         in EMP_INFO with limit of (300)
  cont>         in EMPIDS_LOW with limit of (400);
  ~As: starting map restructure...
  ~As: reads: async 0 synch 0, writes: async 0 synch 0
  ```

  The restructuring can only be eliminated if the original map omitted the OTHERWISE clause and the new map specifies all existing partitions in the same order with the same WITH LIMIT clause.

  Note

  If the ALTER DATABASE ... DROP STORAGE AREA ... CASCADE clause has been used then the ALTER STORAGE MAP should specify the partitions as they currently exist in the storage map, not at they appeared prior to the ALTER DATABASE statement.

- ALTER STORAGE MAP and ALTER INDEX now allow the final WITH LIMIT clause to be dropped from the storage map, effectively changing it to an OTHERWISE partition.
- ALTER STORAGE MAP now supports converting a strictly partitioned storage map to an unrestricted storage map (RANDOMLY ACROSS) without requiring a rebuild of the table. This is permitted if the new map includes all the existing partitions (in any order).

  ```
  SQL> set flags ’stomap_stats’;
  SQL> create table T1 (C1 integer);
  SQL> create storage map S1
  cont>   for T1
  cont>   partitioning is not updatable
  cont>   store using (C1)
  cont>      in JOBS with limit of (100);
  ~As: create storage map "S1"
  ~As: Table "T1" (sys=0, rest=0, tmptbl=0)
  SQL> -- here we drop the restriction (WITH LIMIT clause),
  SQL> -- but use the same area.  No reorganize is needed
  SQL> alter storage map S1 store in JOBS;
  ~As: starting map restructure...
  ~As: reads: async 0 synch 0, writes: async 0 synch 0
  ```

- ALTER STORAGE MAP now clears the PARTITIONING IS NOT UPDATABLE flag when the storage map no longer contains WITH LIMIT clauses. Previously the SHOW STORAGE MAP command in SQL and the RMU/EXTRACT utility would display this erroneous setting. This flag setting is actually ignored at runtime when there are no WITH LIMIT clauses but it can be confusing. Therefore, issue a new ALTER STORAGE MAP after installing this update so that this flag will be cleared.
- CREATE STORAGE MAP now allows the clause PARTITIONING IS UPDATABLE to be specified. In previous versions, this clause would generate the error as shown in this example.

  ```
  SQL> create storage map S1
  cont>   for T1
  cont>   partitioning is updatable
  cont>   store across (JOBS, DEPARTMENTS);
  %RDB-E-NO_META_UPDATE, metadata update failed
  -RDMS-F-MAPBLRMISSING, one or more areas is missing a limitation (WITH LIMIT
  clause)
  ```

  In this case the diagnostic was in error. The syntax is correct and should not have been rejected.
- The SET FLAGS options STOMAP_STATS and INDEX_STATS have been enhanced to display more information during the ALTER STORAGE MAP and ALTER INDEX statements. Example output from these options is shown above.

These problems have been corrected in Oracle Rdb7 Release 7.0.2.1.

3.1.7 Memory Leaks Due to Sorting and Set Flags ’Strategy’

Bug 893877

Memory leaks could be encountered when doing repeated "select count distinct" commands (which does sorting). A leak at the rate of 24 bytes per use of a sort or ’Strategy’ output was observed. For example, after a program had run for quite a while it would appear to have grown by a megabyte for no known reason. For many applications, this may have gone undetected.

A possible workaround is to restart the server on a regular basis, such as weekly.

This problem has been corrected in Oracle Rdb7 Release 7.0.2.1.

3.1.8 DISTINCT With Literal Value Returns Wrong Results on Alpha Platform

Bug 867893

The following query with a DISTINCT clause containing a literal value returns the wrong results on the Alpha platform.

```
SELECT
   distinct 5, TICKET_NUM, AMOUNT, PRODUCT_CODE
 FROM T1, T2
 WHERE
   NEXT_STATION = 0 and
   T1.TICKET_NUM = T2.TICKET_NUMBER ;
               TICKET_NUM                    AMOUNT     PRODUCT_CODE
 5             PI000001       1.000000000000000E+002               0
 5             PI000002       2.000000000000000E+002               0
2 rows selected
```

```
    where E.PRODUCT_CODE should be non-zero.
```

```
The following is the script to reproduce the problem:
```

```
create table T1 (
    TICKET_NUM CHAR (8),
    NEXT_STATION SMALLINT,
    PRODUCT_CODE SMALLINT,
    AMOUNT FLOAT
    );

create table T2 (
    TICKET_NUMBER CHAR (8),
    RECORD_ID     INTEGER
    );

create unique index NDX_T1
    on T1 (
    TICKET_NUM
        asc);

create unique index NDX_T2
    on T2 (
    TICKET_NUMBER
        asc);

insert into T1 (TICKET_NUM, NEXT_STATION, PRODUCT_CODE, AMOUNT)
    value ('PI000001',   0, 1, 100);
insert into T1 (TICKET_NUM, NEXT_STATION, PRODUCT_CODE, AMOUNT)
    VALUE ('PI000002',   0, 2, 200);

insert into T2 (TICKET_NUMBER) VALUE ('PI000001');
insert into T2 (TICKET_NUMBER) VALUE ('PI000002');
```

A workaround is to remove the "5" from the Distinct list in the query, as in the following example.

```
SELECT
   distinct TICKET_NUM, AMOUNT, PRODUCT_CODE
 FROM T1, T2
 WHERE
   NEXT_STATION = 0 and
   T1.TICKET_NUM = T2.TICKET_NUMBER ;
 T1.TICKET_NUM                T1.AMOUNT   T1.PRODUCT_CODE
 PI000001        1.000000000000000E+002                 1
 PI000002        2.000000000000000E+002                 2
2 rows selected
```

This problem has been corrected in Oracle Rdb7 Release 7.0.2.1.

3.1.9 User Process Deleted During Processing of Complex Query

Bugs 491448 and 497343

This symptom can be a result of two different categories of problems.

The first category: in some cases, when complex queries are processed the default executive mode stack size can be exceeded. In earlier versions of Oracle Rdb, this could cause the user process to be deleted. Now instead the following error message may be returned to the user when the executive mode stack is exceeded.

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

If this error is displayed or the user process is deleted, a workaround may be to increase the executive mode stack size by defining the logical RDMS$BIND_EXEC_STACK_SIZE to a sufficent number depending on the complexity of the query, for example 500.

Note

The minimum value allowed for the stack size is 20 (40 for Alpha/VMS). The default stack size is 110 (170 for Alpha/VMS).

The second category involves a query with complex nested CAST...AS INTERVAL expressions which require more executive mode stack space to process. This problem only occurs on OpenVMS Alpha. Although the symptom is the same, unfortunately there is no workaround.

These problems were corrected in Oracle Rdb Version 7.0A. This release note gives additional detail on the minimum and default values for the logical RDMS$BIND_EXEC_STACK_SIZE.

3.2 SQL Errors Fixed

3.2.1 SQL Bugcheck Generated When Illegal Argument Passed to OUT Parameter

Bug 869479

In prior versions of Oracle Rdb, the arguments passed to a CALL statement were not correctly validated in respect to the parameter's calling mode, either IN, OUT or INOUT. This meant that passing an expression, for example a literal, to an INOUT or OUT parameter caused the query compiler to bugcheck.

The following example shows this problem when calling procedure PROC_1 with an incorrect argument.

```
SQL>  create module MODULE_1 language SQL
cont>   procedure PROC_1 (inout :ID_POOL char(8));
cont>   begin
cont>   set :ID_POOL = 'ABCDEFGH';
cont>   end;
cont> end module;
SQL>  create module MODULE_2 language SQL
cont>   procedure PROC_2 ();
cont>   begin
cont>   call PROC_1 ('12345678');      -- Shouldn't pass literal
cont>   end;
cont> end module;
%SQL-I-BUGCHKDMP, generating bugcheck dump file DISK:[DIR]SQLBUGCHK.DMP;
%SQL-F-BUGCHK, There has been a fatal error. Please contact your Oracle support
representative. SQL$SEMMSG - MERGE_PROC_MSG found parsym not in signature
```

The following example shows the problem with a simple CALL statement.

```
SQL> call PROC_1 (null);
%SQL-I-BUGCHKDMP, generating bugcheck dump file DISK:[DIR]SQLBUGCHK.DMP;
%SQL-F-BUGCHK, There has been a fatal error. Please contact your Oracle support
representative. SQL$SEMREQ - Unknown RCV assignment target
```

This problem has been corrected in Oracle Rdb7 Release 7.0.2.1. With this release, SQL now correctly detects that assignment through the CALL statement parameters is illegal. Only procedure parameters (themselves declared as OUT or INOUT), or variables declared as UPDATABLE are allowed to be passed to INOUT and OUT parameters.

3.2.2 Create Database Via Pathname Deadlock

The following command sequence failed with a deadlock using Oracle Rdb7 Release 7.0.2 when the database object did not exist in the repository. This failure did not occur with prior versions.

```
SQL> drop database pathname testdb;
%RDB-E-BAD_DB_FORMAT, testdb does not reference a database known to Rdb
-RMS-E-FNF, file not found
SQL> disconnect all;
SQL> create database filename testdb pathname testdb;
%CDD-E-FSERR, file system error
-RMS-F-DEADLOCK, deadlock detected
```

The workaround to the problem is to exit and reenter the SQL session in order to force the locks to be released.

This problem has been corrected in Oracle Rdb7 Release 7.0.2.1.

3.2.3 Nested SQL Functions May Return the Wrong Results

This release note was inadvertently omitted from previous releases. This problem appeared in Oracle Rdb7 Release 7.0 through 7.0.1.1. When a SQL function was called multiple times in the same expression, it was possible that the intermediate function results would over-write the arguments to the SQL function. This resulted in incorrect results being returned from the outer most function call. This is shown in the following example.

```
create module TEST_NESTED language sql

function PLUS( in :i1 integer, in :i2 integer)
returns integer;
begin
  trace 'i1=',:i1;
  trace 'i2=',:i2;
  return :i1+:i2;
end;

procedure TEST_PLUS();
begin
  declare :i integer;
  set :i = plus( 1000, plus( 100, plus(10, 1)));
  trace 'Result A ', :i;
  trace '';
  set :i = (select plus( 1000, plus( 100, plus(10, 1)))
            from rdb$database);
  trace 'Result B ', :i;
end;

end module;

set flags 'TRACE';
call TEST_PLUS();
```

And here are the results.

```
~Xt: i1=10
~Xt: i2=1
~Xt: i1=10
~Xt: i2=11
~Xt: i1=10
~Xt: i2=21
~Xt: Result a 31

~Xt: i1=10
~Xt: i2=1
~Xt: i1=100
~Xt: i2=11
~Xt: i1=1000
~Xt: i2=111
~Xt: Result b 1111
```

Obviously the correct result is B. This problem occurred due to premature evaluation of the arguments to the SQL function and was only visible if the function was nested as shown in the example. This problem was actually corrected in Oracle Rdb7 Release 7.0.1.2.

3.2.4 DROP INDEX Now an Online Table Operation

DROP INDEX can now be used when other users are processing the table on which the index is defined. This requires that the index has previously been disabled with the ALTER INDEX ... MAINTENANCE IS DISABLED statement.

Once maintenance is disabled for an index, it is no longer used by queries on the table. i.e. it is not used for retrieval, and it is not updated by INSERT, DELETE or UPDATE statements. Therefore, with this release, Rdb has relaxed the requirement of EXCLUSIVE table access for DROP INDEX.

Oracle recommends that the DROP INDEX statement immediately be followed by a COMMIT statement so that all locks on the system metadata be released. Otherwise, access to this and other tables may be stalled waiting for rows locked in the tables RDB$INDICES, RDB$INDEX_SEGMENTS, RDB$STORAGE_MAPS, and RDB$STORAGE_MAP_AREAS.

This change benefits very large databases (VLDB) which have the need to drop indices stored in MIXED format storage areas on large cardinality tables. These indices may take several hours to erase, which in previous versions required taking the table offline from normal processing until the DROP INDEX completed.

Note that indices stored in UNIFORM format storage areas do not take long to DROP due to optimizations which can be made for UNIFORM areas.

```
-- Disable the index maintenance.  This requires exclusive access to the
-- table, but takes a very short time.  This should be done during normal
-- offline maintenance
--
set transaction read write;
alter index TRANSACTION_POSTING_INDEX
    maintenance is disabled;
commit;

-- Once disabled the index can be dropped at any time
--
set transaction read write
    reserving TRANSACTION_POSTING_INDEX shared write;
drop index FOO;
commit;
```

Please note that DROP INDEX will write before image data to the snapshot files (.SNP) if the transaction is started in a mode such as SHARED or PROTECTED. Snapshots can be disabled on the database to avoid the excessive snapshot file I/O during concurrent DROP INDEX operations. Naturally, this may not be possible under normal work loads.

3.2.5 GET DIAGNOSTICS May Return Incorrect RETURNED_SQLCODE

Bug 778720

In some cases, GET DIAGNOSTICS will return an incorrect value for RETURNED_SQLCODE or RETURNED_SQLSTATE. This can occur when a searched UPDATE or searched DELETE is performed (i.e. these statements use the WHERE CURRENT OF clause) and causes a not deferrable constraint to be executed. For a constraint to succeed, it often means that it fetched no matching rows and in this case GET DIAGNOSTICS is returning the SQLCODE or SQLSTATE for the constraint query and not the statement causing the constraint to execute.

The following example shows the incorrect SQLCODE being returned within and after the completion of the FOR loop.

```
SQL> create table substmt (a integer unique, b integer check (b >= 0));
SQL> insert into substmt (a) value (1);
1 row inserted
SQL> set flags 'trace';
SQL>
SQL> begin
cont> declare :sc integer;
cont>
cont> set :sc = 0;
cont> get diagnostics exception 1 :sc = returned_sqlcode;
cont> trace 'no prior statement ', :sc;
cont>
cont> for :substmt_rec
cont>   as each row of cursor sub_cursor for
cont>     select a from substmt
cont> do
cont>     update substmt set b = 0 where current of sub_cursor;
cont>     get diagnostics exception 1 :sc = RETURNED_SQLCODE;
cont>     trace 'after WHERE CURRENT OF UPDATE ', :sc;
cont> end for;
cont>
cont> get diagnostics exception 1 :sc = RETURNED_SQLCODE;
cont> trace 'after FOR loop ', :sc;
cont> end;
~Xt: no prior statement 0
~Xt: after WHERE CURRENT OF UPDATE 100
~Xt: after FOR loop 100
SQL>
```

This problem has been corrected in Oracle Rdb7 Release 7.0.2.1. Actions of constraints no longer affect the returned SQLCODE and SQLSTATE for the query.

3.2.6 Incorrect Values Returned From GET DIAGNOSTICS Statement

Bug 778720

In prior versions of Oracle Rdb, the GET DIAGNOSTICS statement could return incorrect values for ROW_COUNT, RETURNED_SQLSTATE or RETURNED_ SQLCODE. These problems occurred when the FOR cursor loop was used and are listed below.

1. The FOR cursor loop did not set SQLCODE when it failed to fetch any rows. A SQLCODE of 100, or SQLSTATE of '02000' was expected.

   The following example shows this problem.

   ```
   SQL> begin
   cont> declare :rc, :sc integer = 0;
   cont>
   cont> l1:
   cont> for :fs as select * from job_history where employee_id = '00000'
   cont> do
   cont>     trace :fs.job_start;
   cont> end for;
   cont>
   cont> get diagnostics :rc = row_count;
   cont> get diagnostics exception 1 :sc = returned_sqlcode;
   cont> trace :rc, :sc;
   cont>
   cont> end;
   ~Xt: 0          0
   SQL>
   ```

2. If the LEAVE statement was used to exit the FOR cursor loop, then the reported ROW_COUNT would be one less than the actual number of returned rows.

   The following example shows this problem.

   ```
   SQL> attach 'file PERSONNEL';
   SQL> set flags 'trace';
   SQL>
   SQL> begin
   cont> declare :rc, :sc integer = 0;
   cont>
   cont> l1:
   cont> for :fs as select * from job_history where employee_id = '00164'
   cont>      order by job_end
   cont> do
   cont>     trace :fs.job_start;
   cont>     if :fs.job_end is null then leave l1; end if;
   cont> end for;
   cont>
   cont> get diagnostics :rc = row_count;
   cont> get diagnostics exception 1 :sc = returned_sqlcode;
   cont> trace :rc, :sc;
   cont>
   cont> end;
   ~Xt: 1980070500000000
   ~Xt: 1981092100000000
   ~Xt: 1          0
   SQL>
   SQL> begin
   cont> declare :rc, :sc integer = 0;
   cont>
   cont> l1:
   cont> for :fs as select * from job_history where employee_id = '00164'
   cont>      order by job_end
   cont> do
   cont>     trace :fs.job_start;
   cont> end for;
   cont>
   cont> get diagnostics :rc = row_count;
   cont> get diagnostics exception 1 :sc = returned_sqlcode;
   cont> trace :rc, :sc;
   cont>
   cont> end;
   ~Xt: 1980070500000000
   ~Xt: 1981092100000000
   ~Xt: 2          0
   ```

These problems have been corrected in Oracle Rdb7 Release 7.0.2.1.

3.3 Oracle RMU Errors Fixed

3.3.1 RMU/RECOVER/CONFIRM Ignores Confirm Qualifier Unless /RESOLVE is Also Specified

Bug 873871

When recovering two journals with non-contiguous AIJ sequence numbers on the same command line, RMU/RECOVER issues a warning and continues with the journal after the gap. This is expected behavior although database corruption could occur as a result of skipping a journal. The following warning is issued and recovery continues with the next journal.

```
$ rmu/recover/root=testdb AIJ1.AIJBCK,AIJ3.AIJBCK
```

```
%RMU-W-AIJSEQAFT, incorrect AIJ file sequence 2 when 1 was expected
```

Prior versions of Oracle Rdb did not support the /CONFIRM qualifier with /RECOVER unless /RESOLVE was also specified. With this release, /CONFIRM may be used with /RECOVER to enable the user to cancel the recovery if the specified sequence of journals contains a gap.

```
$ rmu/recover/confirm/root=testdb AIJ1.AIJBCK,AIJ3.AIJBCK
```

```
%RMU-W-AIJSEQAFT, incorrect AIJ file sequence 2 when 1 was expected
Do you wish to continue the roll-forward operation [N]:
%RMU-I-AIJRECTRM, AIJ roll-forward operations terminated at user request
```

Another option for recovering multiple journals that will terminate with an error if a gap is encountered is to specify the journals to be recovered on separate command lines. If a journal needed for recovery is skipped, RMU/RECOVER will terminate with the following error:

```
$ rmu/recover/root=testdb AIJ1.AIJBCK
$ rmu/recover/root=testdb AIJ3.AIJBCK
```

```
%RMU-F-AIJNORCVR, recovery of this journal must start with sequence 1
```

This problem has been corrected in Oracle Rdb7 Release 7.0.2.1. With this release, RMU/RECOVER/CONFIRM will prompt the user for confirmation before proceeding to recover journals after a gap is encountered.

3.3.2 RMU Verify Maximum Storage Area Page Invalid

Bug 605131

An RMU VERIFY can return an RMU-W-BADFRAPTR, storage record UNKNOWN, bad fragment chain pointer message if the maximum page number of a storage area has changed since the start of the RMU VERIFY.

The following example shows the error that can be returned if the maximum page number of a storage area has changed since the start of an RMU VERIFY.

```
RMU/VERIFY/ALL SAMPLE_DB
%RMU-I-BGNNDXVER, beginning verification of index KF_IZVBETRAG_SIDX
%RMU-W-BADFRAPTR, area UNIFORM_AREA_010, page 648964, line 18
storage record UNKNOWN, bad fragment chain pointer
expected 1 through 669293, found page number 669955.
%RMU-E-ERRGATFRG, error gathering fragmented record at 21:648964:18
```

This problem was actually corrected in Oracle Rdb7 Release 7.0.2 and inadvertently left out of the Release Notes. Now, when the fragment chain of a storage record is verified and the fragment chain points to a page number which is greater than the maximum page number currently saved for a storage area, the maximum page number for the storage area will be refreshed from the latest value for the database, and the check of the page stored in the fragment chain pointer will be made against the refreshed maximum page number just retrieved from the database.

3.3.3 RMU Analyze Index Used/Available Overflow

Bug 606524

For an RMU ANALYZE INDEX command, the numeric percentage value of Used / Available bytes for index nodes can overflow for large indexes.

The following example shows an overflow value for the percentage value of Used / Available bytes for a large index.

```
RMU/ANALYZE/INDEX SAMPLE_DB SUBSCR_TRANS_INDEX_7B
Index SUBSCR_TRANS_INDEX_7B for relation SUBSCR_TRANSACTIONS
duplicates not allowed
Max Level: 5, Nodes: 1093205, Used/Avail: 1251499453/2167825515
->     (4294967237%),
Keys: 39064727, Records: 37971523
```

This problem was actually corrected in Oracle Rdb7 Release 7.0.2 and inadvertently left out of the Release Notes. The Used/Available index values are now calculated as double float values instead of integer values.

3.4 Row Cache Errors Fixed

3.4.1 RDM$BIND_RCS_LOG_REOPEN_SECS Logical Did Not Work

The RDM$BIND_RCS_LOG_REOPEN_SECS logical did not work even though it was defined properly. However, the RDM$BIND_RCS_LOG_REOPEN_SIZE logical, which has a similar effect of controlling the size of the log file, was implemented correctly and could be used as a potential workaround for avoiding very large log files.

This problem has been corrected in Oracle Rdb7 Release 7.0.2.1.

3.4.2 File Access Errors to RDC Files Cause Checkpoints to be Directed to Database

Previously, when using the Oracle Rdb Row Cache feature, if the Row Cache Server (RCS) process was unable to perform a checkpoint operation to the backing store (.RDC) files, the RCS process would fail and the database would be shut down.

If, for example, the disk used for the backing store files became full, the RCS process would abort due to the failure to extend the backing store file during a checkpoint operation.

This situation has been improved. The RDC process now reverts to checkpointing to the database when it is unable to write to a backing store file during a checkpoint operation.

The system operator is notified with a message of “Problem accessing an RDC file for Oracle Rdb database ’db’; RCS checkpointing to database instead ”. For each new checkpoint operation, the RCS process will attempt to write to the backing store files. If the disk problem has corrected (by additional disk space being freed, for example), the checkpoint will continue correctly. Otherwise, it will revert back to the database until the next checkpoint.

This problem was actually corrected in Oracle Rdb7 Release 7.0.2 but the release note was inadvertently left out.

4 Documentation Corrections

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

4.1 Documentation Corrections

4.1.1 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*.

4.1.2 Oracle Rdb Logical Names

The *Oracle Rdb7 Guide to Database Performance and Tuning* contains a table in Chapter 2 summarizing the Oracle Rdb logical names 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.

4.1.3 Waiting for Client Lock Message

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

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

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

The leftmost value seen in the hexadecimal output contains the id of the object. The id is described below for 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 which 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 a 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.

4.1.4 Documentation Error in the 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 below:

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

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

4.1.5 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 by 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.

4.1.6 SET FLAGS Option INTERNALS Not Described

The *Oracle Rdb7 SQL Reference Manual* does not described 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 flags settings are equivalent to defining the RDMS$DEBUG_FLAGS as

"ISn" and shows the strategy used by the triggers actions on the AFTER DELETE trigger on EMPLOYEES.

```
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
```

4.1.7 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 OpenVMS logical RDMS$VALIDATE_ROUTINE, or the Digital UNIX environment variable SQL_VALIDATE_ROUTINE described in the *Oracle Rdb7 Guide to Database Performance and Tuning*.

This example shows the revalidation 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 set a 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.

4.1.8 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. Below is one example where the RDBSERVER.COM procedure references SYS$COMMON:<SYSEXE> and SYS$COMMON:[SYSEXE] rather than SYS$SYSTEM.

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

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

The *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
```

4.1.9 Undocumented SET Commands and Language Options

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

4.1.9.1 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 which 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 in this case is 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.

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

PROCEDURE S_TXN (SQLCODE);
SET TRANSACTION READ WRITE;

PROCEDURE C_TXN (SQLCODE);
COMMIT;
```

4.1.9.2 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 which 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 transaction starting 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).

```
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;
```

4.1.10 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 Rdb which have supported 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 to store. 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
```

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

When global buffers are used, the number of buffers used by the AIJ Log Roll-forward Server is determined as follows:

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

The /BUFFER qualifier now enforces a minimum of 256 buffers for the AIJ Log Roll-forward Server. The maximum number of buffers allowed is still 524288 buffers.

5
Known Problems and Restrictions

This chapter describes problems, restrictions, and workarounds known to exist in Oracle Rdb7 Release 7.0.3.

5.0.1 SQL Uses READ WRITE Transactions Against STANDBY Database

In this release and prior releases of Oracle Rdb7, the SQL interface, in particular interactive and dynamic SQL, will attempt to execute a READ WRITE transaction on a database which is currently a Standby database. This type of database is considered READ ONLY and starting a READ WRITE transaction fails. This problem may cause connections from ODBC, SQL*Net for Rdb and SQL/Services to fail.

A workaround for this problem is to restore the standby database using the RMU/RESTORE qualifier /TRANSACTION_MODE=READ_ONLY which permits only READ ONLY transactions on the database. Current releases of SQL do detect this setting and so will then operate correctly. When the standby is required as the master (following a system failure), the ALTER DATABASE ... ALTER TRANSACTION MODE (READ WRITE) should be used to enable READ WRITE transactions for the database.

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

5.0.2 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, drop the affected index and re-create it under Oracle Rdb7 Release 7.0.1.3 or later.

5.0.3 DBAPack for Windows 3.1 is Deprecated

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

5.0.4 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 rules shown below.

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.

  ```
  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.

  ```
  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.

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

Any parameter which 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.

  ```
  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 hand side of the assignment in a SET statement or SET clause of an UPDATE statement.

  ```
  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.

  ```
  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.

  ```
  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.

  ```
  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/ELSE. 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.

5.0.5 DROP TABLE CASCADE Will Result 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 drop the storage map, and then drop the table using the cascade option. The following example shows the workaround. The SHOW statement indicates that the table, index, and storage map were dropped.

```
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 Rdb7.

5.0.6 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 World-Wide Support.

5.0.7 Interruptions Possible 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 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.

5.0.8 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 effected; 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.

5.0.9 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 LRS (Log Rollforward Server) will wait until the read-only transaction(s) commits and then replication will continue.

This is an existing restriction of the Hot Standby software. This release note is intended to compliment the Hot Standby documentation.

5.0.10 Using the SYS$LIBRARY:SQL_FUNCTIONS70.SQL Oracle Functions Script

All OpenVMS platforms.

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 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 setting shown in the following example:

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

In a multi-version environment, use the setting shown in the following example:

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

5.0.11 DECC and Use of the /STANDARD Switch

Bug 394451.

All OpenVMS platforms.

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 DECC was put into the product with V6.0 and this note is meant to clarify that support, not to indicate a change. It's 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 Rdb SQL program. Assume a symbol SQL$PRE has been defined and DECC 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 VAXC. Those are also not supported.

5.0.12 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 a query and index creation operations. Defining the logical name RDMS$DEBUG_FLAGS to "RS" can help determine when Oracle Rdb sort operations are occurring and to display the sort keys and statistics.

Oracle Rdb includes its own copy of the OpenVMS SORT32 code within the Oracle Rdb images and does not generally call the routines in the OpenVMS run-time library. A copy of the SORT32 code is used to provide stability between versions of Oracle Rdb and OpenVMS and because Oracle Rdb calls the sort routines from executive processor mode which is difficult to do using the SORT32 sharable image. Database import and RMU load operations do 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 processes' quotas or parameters when allocating memory. The effects of WSQUOTA and WSEXTENT are indirect. At the beginning of each sort operation, the sort code attempts to adjust the process working set to the maximum possible size using the $ADJWSL system service specifying a requested working set limit of %X7FFFFFFF pages (the maximum possible). Sort 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's working set is limited by the working set quota (WSQUOTA) parameter and the working set extent (WSEXTENT) parameter is a much larger value, the first call to the sort routines can cause many page faults as the working set grows. Using a value of WSEXTENT that is closer to WSQUOTA can help reduce the impact of this case.

With some OpenVMS versions, AUTOGEN sets the SYSGEN parameter PQL_MWSEXTENT equal to the WSMAX parameter. This means that all processes on the system end up with WSEXTENT the same as WSMAX. Since that might be quite high, sorting might result in excessive page faulting. You may want to explicitly set PQL_MWSEXTENT to a lower value if this is the case on your system.

Sort work files are another factor to consider when tuning Oracle Rdb sort operations. When the operation can not 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 ten 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.

5.0.13 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.

5.0.14 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.

5.0.15 RMU Backup Operations and Tape Drive Types

When using more than one tape drive for an RMU backup operation, all of 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 Rdb RMU attempts to prevent using different tape drive densities during a backup operation, but is not able to detect all invalid cases and expects that all tape drives for a backup are of the same type.

As long as all of the tapes used during a backup operation can be read by the same type of tape drive during a restore operation, the backup is likely valid. This may be the case, for example, when using a TA90 and a TA90E.

Oracle recommends that, on a regular basis, you test your backup and recovery procedures and environment using a test system. You should restore the database(s) and then recover 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 Rdb RMU Reference Manual* for additional information about Oracle Rdb backup and restore operations.

5.0.16 Use of RDB from Shared Images

All OpenVMS platforms.

Bug 470946.

If code in the image initialization routine of a shared image makes any calls into RDB, through SQL or any other means, access violations or other unexpected behavior may occur if 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 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 RDB shared images **last** in the linker options file.

5.0.17 Interactive SQL Command Line Editor Rejects Eight Bit Characters

Digital UNIX platform.

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.

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

To quote a character using the command line editor, type Ctrl/V before each character to be quoted.

5.0.18 Restriction Added for CREATE STORAGE MAP on Table with Data

Oracle Rdb7 added support which allows a storage map to be added to an existing table which contains data. The restrictions listed for Rdb7 were:

- The storage map must be a simple map which 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 may not change THRESHOLDS or COMPRESSION for the table, nor can it use the PLACEMENT VIA INDEX clause. It may only contain one area and may not be vertically partitioned. This new map simply describes the mapping as it exists by default for the table.

This version 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
```

5.0.19 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 drop 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 below, 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.

5.0.20 Monitor ENQLM Minimum Increased to 32767

All OpenVMS platforms.

In previous versions, the Oracle Rdb7 monitor process (RDMMON) was created with a minimum lock limit (ENQLM) of 8192 locks. This minimum has been increased to 32767 locks (the OpenVMS maximum value).

5.0.21 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 disconnect 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);
```

5.0.22 RMU Convert Command and System Tables

When the RMU Convert command converts a database from a previous version to Oracle Rdb7 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.

5.0.23 Converting Single-File Databases

Because of a substantial increase in the database root file information for V7.0, you should ensure that you have adequate disk space before you use the RMU Convert command with single-file databases and V7.0 or higher.

The size of the database root file of any given database 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.

5.0.24 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, 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.

5.0.25 Restriction on Tape Usage for Digital UNIX V3.2

Digital UNIX platforms.

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, you may have encountered the following situation:

If an error is detected by Digital UNIX during the open operation of the tape device, 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.

5.0.26 Support for Single-File Databases to Be Dropped in a Future Release

Oracle Rdb7 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 Rdb7. At that time, Oracle Rdb7 will provide the means to easily convert single-file databases to multifile databases.

Oracle Rdb7 recommends that users with single-file databases perform the following actions:

- Use the Oracle RMU commands, such as Backup and Restore, to make copies, backup, 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 V6.1 and V7.0.

5.0.27 DECdtm Log Stalls

All OpenVMS platforms.

Resource managers using the DECdtm services sometimes suddenly stop being able to commit transactions. The systems have been running fine for some period of time, but suddenly they stop. 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 to this problem. One workaround, for OpenVMS V5.5-x, is provided here.

On the affected node, and while the log stall is in progress, perform 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 Digital representative for information about patches to DECdtm.

5.0.28 You Cannot Run Distributed Transactions on Systems with DECnet/OSI and OpenVMS Alpha Version 6.1 or OpenVMS VAX Version 6.0

All OpenVMS platforms.

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 V6.2 or higher, you can run Oracle Rdb7 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*.

5.0.29 Multiblock Page Writes May Require Restore Operation

All OpenVMS platforms.

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 Rdb7 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.

5.0.30 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 V4.1 and earlier, a FINISH statement), which also fails with an RDB-E-IO_ ERROR error message.

5.0.31 Replication Option Copy Processes Do Not Process Database Pages Ahead of an Application

All OpenVMS platforms.

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 V4.0 and higher. Oracle Rdb7 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 Rdb7 verb is an Oracle Rdb7 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.

5.0.32 SQL Does Not Display Storage Map Definition After Cascading Delete of Storage Area

When you drop 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.

5.0.33 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.)

5.0.34 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 Rdb7 system tables RDB$RELATIONS and RDB$RELATION_FIELDS exceed the limit of rows.

  Oracle Rdb7 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 Rdb7 system tables as part of query processing.

5.0.35 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 Rdb7 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 Rdb7 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 ten parallel process index creations (Index1, Index2, . . . Index10) and one process with ten sequential index creations (All10). In this example, global buffers are enabled and the number of buffers is 500. The longest time for a parallel index creation is Index7 with an elapsed time of 00:02:34.64, compared to creating ten indexes sequentially with an elapsed time of 00:03:26.66. The longest single parallel create index elapsed time is shorter than the elapsed time of creating all ten of the indexes serially.

Index Create Job	Elapsed Time
Index1	00:02:22.50
Index2	00:01:57.94
Index3	00:02:06.27
Index4	00:01:34.53
Index5	00:01:51.96
Index6	00:01:27.57
Index7	00:02:34.64
Index8	00:01:40.56
Index9	00:01:34.43
Index10	00:01:47.44
All10	00:03:26.66

5.0.36 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.

5.0.37 Nested Correlated Subquery Outer References Incorrect

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 nth - 1 row data (usually NULL for row 1) when it should have been using the nth row data.

This problem has existed in various forms for all previous versions of Oracle Rdb7, but only appears in V6.1 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 will be corrected in a future release of Oracle Rdb.

5.0.38 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 Rdb7 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 Rdb7 to take a data snapshot, the data read and cached may be stale by the time the cursor fetches the data.

  For example, user 1 opens a cursor and commits the transaction. User 2 deletes rows read by user 1 (this is possible because the read locks are released). It is possible for user 1 to report data now deleted and committed.

- If the access strategy uses indexes that allow duplicates, updates to the duplicates chain may cause rows to be skipped, or even revisited.

  Oracle Rdb7 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 Rdb7 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 Rdb7.

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.

5.0.39 INCLUDE SQLDA2 Statement Is Not Supported for SQL Precompiler for PL/I in Oracle Rdb Release 5.0 or Higher

All OpenVMS platforms.
The SQL statement INCLUDE SQLDA2 is not supported for use with the PL/I precompiler in Oracle Rdb7 V5.0 or higher.

There is no workaround. This problem will be fixed in a future version of Oracle Rdb7.

5.0.40 SQL Pascal Precompiler Processes ARRAY OF RECORD Declarations Incorrectly

All OpenVMS platforms.
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.

To avoid the problem, 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;
```

5.0.41 RMU Parallel Backup Command Not Supported for Use with SLS

All OpenVMS platforms.

The RMU Parallel Backup command is not supported for use with the Storage Library System (SLS) for OpenVMS.

5.0.42 Oracle RMU Commands Pause During Tape Rewind

Digital UNIX platforms.

For Oracle Rdb V6.1 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.

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

Digital UNIX platforms.

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 for Oracle Rdb7

This section describes known problems and restrictions in Oracle CDD/Repository V7.0 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 Version of Oracle Rdb	Minimum Version of Oracle CDD/Repository	Support
CASE, NULLIF, and COALESCE expressions	V6.0	V6.1	Implicit
CAST function	V4.1	V7.0	Explicit
Character data types to support character sets	V4.2	V6.1	Implicit
Collating sequences	V3.1	V6.1	Explicit
Constraints (PRIMARY KEY, UNIQUE, NOT NULL, CHECK, FOREIGN KEY)	V3.1	V5.2	Explicit
CURRENT_DATE, CURRENT_TIME, and CURRENT_TIMESTAMP functions	V4.1	V7.0	Explicit
CURRENT_USER, SESSION_USER, SYSTEM_USER functions	V6.0	V7.0	Explict
Date arithmetic	V4.1	V6.1	Pass-through
DATE ANSI, TIME, TIMESTAMP, and INTERVAL data types	V4.1	V6.1	Explicit
Delimited identifiers	V4.2	V6.1[1]	Explicit
External functions	V6.0	V6.1	Pass-through
External procedures	V7.0	V6.1	Pass-through
EXTRACT, CHAR_LENGTH, and OCTET_LENGTH functions	V4.1	V6.1	Explicit
GRANT/REVOKE privileges	V4.0	V5.0 accepts but does not store information	Pass-through
Indexes	V1.0	V5.2	Explicit
INTEGRATE DOMAIN	V6.1	V6.1	Explicit
INTEGRATE TABLE	V6.1	V6.1	Explicit
Logical area thresholds for storage maps and indexes	V4.1	V5.2	Pass-through
Multinational character set	V3.1	V4.0	Explicit
Multiversion environment (multiple Rdb versions)	V4.1	V5.1	Explicit
NULL keyword	V2.2	V7.0	Explicit
Oracle7 compatibility functions, such as CONCAT, CONVERT, DECODE, and SYSDATE	V7.0	V7.0	Explicit
Outer joins, derived tables	V6.0	V7.0	Pass-through
Query outlines	V6.0	V6.1	Pass-through
Storage map definitions correctly restored	V3.0	V5.1	Explicit

[1]The repository does not preserve the distinction between uppercase and lowercase identifiers. If you use delimited identifiers with Oracle Rdb7, 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 Version of Oracle Rdb	Minimum Version of Oracle CDD/Repository	Support
Stored functions	V7.0	V6.1	Pass-through
Stored procedures	V6.0	V6.1	Pass-through
SUBSTRING function	V4.0	V7.0 supports all features V5.0 supports all but V4.2 MIA features [2]	Explicit
Temporary tables	V7.0	V6.1	Pass-through
Triggers	V3.1	V5.2	Pass-through
TRUNCATE TABLE	V7.0	V6.1	Pass-through
TRIM and POSITION functions	V6.1	V7.0	Explicit
UPPER, LOWER, TRANSLATE functions	V4.2	V7.0	Explicit
USER function	V2.2	V7.0	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 Rdb7 V7.0 and earlier. This problem will be corrected in a future release of Oracle Rdb7.

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

OpenVMS VAX platforms.

Oracle Rdb7 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 Rdb7.

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 specially 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 platforms.

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 platforms.

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
```