Oracle® Rdb for OpenVMS

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Oracle® Rdb for OpenVMS

New Features Document

Release 7.2.x.x

June 2011

Oracle Rdb New Features, Release 7.2.x.x for OpenVMS

Contents

Preface

Purpose of This Manual

This manual contains the New Features Chapters for Oracle Rdb Release 7.2.5.0 and prior Rdb 7.2.x.x releases.

Document Structure

This manual consists of the following chapters:

Chapter 1	Describes enhancements introduced in Oracle Rdb Release 7.2.5.0
Chapter 2	Describes enhancements introduced in Oracle Rdb Release 7.2.4.2
Chapter 3	Describes enhancements introduced in Oracle Rdb Release 7.2.4.1
Chapter 4	Describes enhancements introduced in Oracle Rdb Release 7.2.4.0
Chapter 5	Describes enhancements introduced in Oracle Rdb Release 7.2.3.5.
Chapter 6	Describes enhancements introduced in Oracle Rdb Release 7.2.3.2.
Chapter 7	Describes enhancements introduced in Oracle Rdb Release 7.2.3.0.
Chapter 8	Describes enhancements introduced in Oracle Rdb Release 7.2.2.0.
Chapter 9	Describes enhancements introduced in Oracle Rdb Release 7.2.1.4.
Chapter 10	Describes enhancements introduced in Oracle Rdb Release 7.2.1.3.
Chapter 11	Describes enhancements introduced in Oracle Rdb Release 7.2.1.2.
Chapter 12	Describes enhancements introduced in Oracle Rdb Release 7.2.1.0.
Chapter 13	Describes enhancements introduced in Oracle Rdb Release 7.2.0.2.
Chapter 14	Describes enhancements introduced in Oracle Rdb Release 7.2.0.1.
Chapter 15	Describes enhancements introduced in Oracle Rdb Release 7.2.

Chapter 1
Enhancements And Changes Provided in Oracle Rdb Release 7.2.5.0

1.1 Enhancements And Changes Provided in Oracle Rdb Release 7.2.5.0

1.1.1 RMU /SHOW STATISTICS /ROWS= and /COLUMNS= Feature

Previously, it was not possible to use the RMU /SHOW STATISTICS and specify the number of display rows and columns. The values would default to the user's display or, in the case of non−interactive mode, 132 columns and 66 rows.

This problem has been corrected in Oracle Rdb Release 7.2.5. RMU /SHOW STATISTICS includes the new qualifiers /ROWS=n and /COLUMNS=n to allow the user to specify the desired number of display rows and columns. The existing minimum and maximum limits apply as enforced by the SMG run time library or the RMU /SHOW STATISTICS utility.

1.1.2 New LIMIT Clauses Implemented for the CREATE and ALTER PROFILE Statement

In prior versions of Oracle Rdb, the LIMIT clauses of the CREATE PROFILE statement and the ALTER PROFILE statement were incomplete. These clauses are now active in this release of Rdb.

The following information replaces the description of these clauses in the current Oracle Rdb SQL Reference Manual, Volume 2 for the CREATE PROFILE Statement.

Arguments

- LIMIT CPU TIME
  LIMIT CPU TIME sets the maximum CPU time that can be used by the query compiler. The keyword DEFAULT indicates that no value is defined by this profile and is equivalent to NO LIMIT CPU TIME.
  If a numeric value or the keyword UNLIMITED is specified then this value will be used even when the SET QUERY LIMIT CPU TIME statement is present in the session, or when the logical name RDMS$BIND_QG_CPU_TIMEOUT is defined.
  NO LIMIT CPU TIME is the default. Units can be specified as seconds or minutes.
- LIMIT TIME
  LIMIT TIME sets the maximum elapsed time that can be used by the query compiler. The keyword DEFAULT indicates that no value is defined by this profile and is equivalent to NO LIMIT TIME.
  If a numeric value or the keyword UNLIMITED is specified then this value will be used even when the SET QUERY LIMIT TIME statement is present in the session, or when the logical name RDMS$BIND_QG_TIMEOUT is defined.
  NO LIMIT TIME is the default. Units can be specified as seconds or minutes.
- LIMIT ROWS
  LIMIT ROWS sets the maximum number of rows that can be returned by a query started by the user. The keyword DEFAULT indicates that no value is defined by this profile and is equivalent to NO LIMIT ROWS.
  If a numeric value or the keyword UNLIMITED is specified then this value will be used even when

the SET QUERY LIMIT ROWS statement is present in the session, or when the logical name RDMS$BIND_QG_REC_LIMIT is defined.
NO LIMIT ROWS is the default.

Examples

This example shows the use of the LIMIT clauses to set boundaries for standard database users.

```
SQL> create profile STANDARD_USER
cont>    limit rows 10000
cont>    limit time 10 minutes
cont>    limit cpu time 20 seconds;
SQL> show profile STANDARD_USER;
     STANDARD_USER
     Limit rows 10000
     Limit time 10 minutes
     Limit CPU time 20 seconds
SQL> alter profile STANDARD_USER
cont>    limit time 60 minutes;
```

Usage Notes

- The logical names RDMS$BIND_QG_REC_LIMIT, RDMS$BIND_QG_TIMEOUT, and RDMS$BIND_QG_CPU_TIMEOUT establish the process defaults for the query limit.
- The command SET QUERY LIMIT establishes the session default (unless already set by the query governor logical names).
- The profile LIMIT will either use these established defaults (LIMIT ... DEFAULT or NO LIMIT) or override them (LIMIT ... UNLIMITED or specified value).

1.1.3 Use of RMS MBC Larger Than 127

This release of Oracle Rdb takes advantage of OpenVMS enhancements permitting values of the RMS Multi Block Count (MBC) parameter to be up to 255 blocks (the prior limit was 127 blocks). With this change, some disk−based file read and write operations performed by Oracle Rdb may require half of the IO resources as compared with prior releases by allowing RMS to do larger IO transfers.

Oracle Rdb now anticipates that OpenVMS patch(es) have been installed that support using an RMS Multi Block Count (MBC) parameter larger than 127 blocks. Oracle Rdb will first attempt to use a larger value and if an RMS$_MBC error is returned from the SYS$CONNECT call, a second attempt is made with a RMS Multi Block Count (MBC) parameter of less than 128.

In order to receive the IO performance improvements available when accessing sequential files when using an RMS Multi Block Count (MBC) parameter larger than 127 blocks, the following patches (or their subsequent replacements) are required to be installed:

- VMS84I_UPDATE−V0400 or later
- VMS84A_UPDATE−V0400 or later
- VMS831H1I_SYS−V1200 or later
- VMS83I_SYS−V1500 or later
- VMS83A_SYS−V1800 or later

Oracle recommends installing OpenVMS patches that permit values of the RMS Multi Block Count (MBC) parameter to be up to 255 blocks for best performance and functionality.

1.1.4 New Optimizations for the LIKE Predicate

Bugs 3516321 and 9931047

This release of Oracle Rdb will try to rewrite the LIKE predicate when the LIKE pattern is a string literal. This enhancement allows Rdb to simplify some LIKE expressions resulting in reduced I/O and CPU time required to satisfy these queries.

- When the LIKE pattern is only the "%" character (which matches one or more characters in the source string) then this will be replaced with an OCTET_LENGTH (...) >= 0 condition that does not require any pattern matching. Note that a string of "%" wildcards, such as "%%%%%", behaves in the same way as a single "%" character. Therefore, the same optimization is applied when any number of trailing % wildcards are detected.

```
SQL> select count(*)
cont>  from employees
cont>  where middle_initial like '%';
Tables:
  0 = EMPLOYEES
Aggregate: 0:COUNT (*)
Conjunct: OCTET_LENGTH (0.MIDDLE_INITIAL) >= 0
Get     Retrieval sequentially of relation 0:EMPLOYEES

          64
1 row selected
SQL>
```

- When the LIKE pattern is only a series of "_" characters (one or more) and the pattern length is the same size as the CHAR source expression, or the VARCHAR source expression matches the length of the pattern, then this will be replaced with an OCTET_LENGTH (...) = n condition that does not require any pattern matching.
  The following example shows a LIKE match against a fixed length CHAR column STATUS_NAME.

```
SQL> select status_name
cont>  from work_status
cont>  where status_name like '________';
Tables:
  0 = WORK_STATUS
Conjunct: OCTET_LENGTH (0.STATUS_NAME) = 8
Get     Retrieval sequentially of relation 0:WORK_STATUS
 STATUS_NAME
 INACTIVE
 ACTIVE
 ACTIVE
3 rows selected
SQL>
```

  The following example shows the string matching the variable length VARCHAR value which is six octets in length.

```
SQL> -- Find six character LAST_NAME values
SQL> select first_name, middle_initial, last_name
```

```
cont>  from candidates
cont>  where last_name like '______';
Tables:
  0 = CANDIDATES
Conjunct: OCTET_LENGTH (0.LAST_NAME) = 6
Get     Retrieval sequentially of relation 0:CANDIDATES
 FIRST_NAME   MIDDLE_INITIAL   LAST_NAME
 Oscar        M.               Wilson
1 row selected
SQL>
```

- When the LIKE pattern starts with a string not containing wildcard characters ("%", "_", or program supplied escape character) but followed by only the "%" wildcard, then Oracle Rdb can replace this with the STARTING WITH operator.

```
SQL> select employee_id, first_name, last_name
cont>  from employees
cont>  where last_name like 'Smith%';
Tables:
  0 = EMPLOYEES
Leaf#01 FFirst 0:EMPLOYEES Card=100
  Bool: 0.LAST_NAME STARTING WITH 'Smith'
  BgrNdx1 EMPLOYEES_LAST_NAME [1:1] Fan=14
    Keys: 0.LAST_NAME STARTING WITH 'Smith'
 EMPLOYEE_ID   FIRST_NAME   LAST_NAME
 00165         Terry        Smith
 00209         Roger        Smith
2 rows selected
SQL>
```

Note

Oracle Rdb has for some time made use of pattern prefix strings to start index scans when possible. This feature is now highlighted in the STRATEGY output by displaying "(Starting With)" after the LIKE predicate for the index keys. This optimization is applied to string literals, column references, host variables, and other string expressions.

```
SQL> declare :patt varchar(10) = 'Smit%';
SQL> select employee_id, first_name, last_name
cont>  from employees
cont>  where last_name like :patt;
Tables:
  0 = EMPLOYEES
Leaf#01 FFirst 0:EMPLOYEES Card=100
  Bool: 0.LAST_NAME LIKE <var0>
  BgrNdx1 EMPL_LAST_NAME [1:1] Fan=12
    Keys: 0.LAST_NAME LIKE <var0> (Starting With)
    Bool: 0.LAST_NAME LIKE <var0>
 EMPLOYEE_ID   FIRST_NAME   LAST_NAME
 00165         Terry        Smith
 00209         Roger        Smith
2 rows selected
SQL>
```

Conversely, if the LIKE pattern string literal has leading wildcards, then this optimization will be disabled since we know during query compile that such an optimization would provide no benefit and thus we save CPU time for such queries.

```
SQL> select employee_id, first_name, last_name
cont>  from employees
cont>  where last_name like '%Smith%';
Tables:
  0 = EMPLOYEES
Leaf#01 FFirst 0:EMPLOYEES Card=100
  Bool: 0.LAST_NAME LIKE '%Smith%'
  BgrNdx1 EMPL_LAST_NAME [0:0] Fan=12
    Keys: 0.LAST_NAME LIKE '%Smith%'
    Bool: 0.LAST_NAME LIKE '%Smith%'
 EMPLOYEE_ID   FIRST_NAME   LAST_NAME
 00165         Terry        Smith
 00209         Roger        Smith
2 rows selected
SQL>
```

- When the LIKE pattern does not contain any wildcard characters, then the LIKE may be converted to an equals condition with a restriction on the length. In general, this produces better query strategies as it allows leading index segments to be matched, where in prior versions this LIKE reference would terminate the partial key construction.

```
SQL> select count (*)
cont>  from employees
cont>  where sex like 'F';
Tables:
  0 = EMPLOYEES
Aggregate: 0:COUNT (*)
Conjunct: 0.SEX = 'F'
Get     Retrieval by index of relation 0:EMPLOYEES
  Index name  EMPLOYEES_LAST_NAME [0:0]

                   35
1 row selected
SQL>
```

This example uses a CANDIDATES table where the LAST_NAME column is an indexed VARCHAR type. Observe that the LIKE was transformed to an equals condition that enabled a direct index lookup.

```
SQL> select last_name, first_name
cont> from CANDIDATES
cont>  where last_name like 'Wilson';
Tables:
  0 = CANDIDATES
Leaf#01 FFirst 0:CANDIDATES Card=3
  Bool: (0.LAST_NAME = 'Wilson') AND (OCTET_LENGTH (0.LAST_NAME) = 6)
  BgrNdx1 CAND_LAST_NAME [1:1] Fan=12
    Keys: 0.LAST_NAME = 'Wilson'
 LAST_NAME        FIRST_NAME
 Wilson           Oscar
1 row selected
```

The RDMS$SET_FLAGS logical name or the SQL SET FLAGS statement can disable this default behavior by using the 'NOREWRITE(LIKE)' keywords. The default is SET FLAGS 'REWRITE(LIKE)'.

1.1.5 Additional Database Storage Area Checks

In rare cases generally involving concealed logical names or in a cluster environment, it was possible to construct cases where two physical databases could access the same storage area file in an uncoordinated fashion. This access could lead to data corruption.

The steps required for this type of uncoordinated access would include a database restore or copy likely in a cluster environment. Because the databases shared the same original creation time stamp, so too would the storage area. Oracle Rdb would not detect that the storage areas were for physically separate databases.

This problem has been corrected in Oracle Rdb Release 7.2.5. Oracle Rdb now stores a time stamp of the physical database creation (via copy, restore or create) in the database root file and each storage area file. Each new access to a storage area compares the time stamp of the database root file and the storage area file to further ensure that they are part of the same physical database. If a mismatch is detected, the message INVDBSFIL "inconsistent storage area file" is signaled.

1.1.6 New Optimizations for the STARTING WITH Predicate

This release of Oracle Rdb will try to rewrite the STARTING WITH predicate when the STARTING WITH string is a literal. This enhancement allows Rdb to simplify some STARTING WITH expressions resulting in better index use and usually reducing I/O and CPU time requirements for these queries.

- When the STARTING WITH string is a zero length literal value then it will cause the STARTING WITH to match all non−NULL values. This can cause an unnecessary favoring of an index lookup which consumes CPU time with little advantage to the application.
- When the STARTING WITH string literal is the exact length of the source expression then complete non−NULL values are selected and Rdb can replace the STARTING WITH with an equality (=) predicate.

```
SQL> select last_name, first_name
cont> from employees
cont> where last_name starting with 'Smith         ';
Tables:
  0 = EMPLOYEES
Leaf#01 FFirst 0:EMPLOYEES Card=100
  Bool: 0.LAST_NAME = 'Smith         '
  BgrNdx1 EMPLOYEES_LAST_NAME [1:1] Fan=12
    Keys: 0.LAST_NAME = 'Smith         '
 LAST_NAME        FIRST_NAME
 Smith            Terry
 Smith            Roger
2 rows selected
```

The RDMS$SET_FLAGS logical name or the SQL SET FLAGS statement can disable this default behavior by using the 'NOREWRITE(STARTING_WITH)' keywords. The default is SET FLAGS 'REWRITE(STARTING_WITH)';

1.1.7 New Optimizations for the CONTAINING Predicate

This release of Oracle Rdb will try to rewrite the CONTAINING predicate when the CONTAINING string is a literal value. This enhancement allows Rdb to simplify some CONTAINING expressions resulting in better

index use and usually reducing I/O and CPU time requirements for these queries.

- When the CONTAINING string is a zero length literal value then it will cause the CONTAINING to match all non−NULL values. This can consume CPU time with little advantage to the application. This is replaced by a semantically equivalent OCTET_LENGTH function reference.

The RDMS$SET_FLAGS logical name or the SQL SET FLAGS statement can disable this default behavior by using the 'NOREWRITE(CONTAINING)' keywords. The default is SET FLAGS 'REWRITE(CONTAINING)'.

1.1.8 Monitor Memory Management Enhancements

Previously, the Oracle Rdb Monitor (RDMMON) process would map each database global (TROOT) section into P0 virtual address space. This could, in some cases, consume a significant portion of the 1GB available space and could also result in the virtual address space becoming sufficiently fragmented such that the monitor would be unable to open a database.

As a possible workaround, the monitor process can be restarted.

The impact of this virtual memory fragmentation has been somewhat reduced. The Oracle Rdb Monitor (RDMMON) process now maps database global sections (those that use SHARED MEMORY IS PROCESS or SHARED MEMORY IS PROCESS RESIDENT) into 64−bit P2 virtual address space. In addition, on OpenVMS Integrity Server systems, the executable code of the Oracle Rdb Monitor (RDMMON) process is mapped into 64−bit P2 virtual address space further reducing the amount of P0 virtual address space consumed.

1.1.9 Average Transaction Duration Display Precision Increased

As systems and applications become faster, it is more common for transaction durations of less than .0000 seconds to be performed. Currently, it is not possible to accurately determine, with the RMU /SHOW STATISTICS utility, the average duration of such rapid transactions. This condition has been addressed.

The average transaction duration value on the "Transaction Duration" display of the RMU /SHOW STATISTICS utility has been increased in precision from 4 to 6 fractional digits and the output display slightly modified to fit into an 80 column display, as in the following example.

```
Node: RDBTUK (1/1/1) Oracle Rdb V7.2−50 Perf. Monitor 29−SEP−2010 23:10:29.65
Rate: 3.00 Seconds     Transaction Duration (Total)      Elapsed: 00:00:26.77
Page: 1 of 1          DISK$TUKWILA7:[DB.V72]FOO.RDB;1            Mode: Online
−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
Total transaction count:            771
Seconds Tx.Count:   %  #Complete:   % #Incomplete:   %
 0−< 1:       773 100%         773 100%            0   0% <−avg=0.000192 95%=0.0
 1−< 2:         0   0%           0   0%            0   0%
 2−< 3:         0   0%           0   0%            0   0%
 3−< 4:         0   0%           0   0%            0   0%
 4−< 5:         0   0%           0   0%            0   0%
 5−< 6:         0   0%           0   0%            0   0%
 6−< 7:         0   0%           0   0%            0   0%
 7−< 8:         0   0%           0   0%            0   0%
 8−< 9:         0   0%           0   0%            0   0%
```

```
 9-<10:         0   0%          0   0%           0   0%
 10+++:         0   0%          0   0%           0   0%
```

1.1.10 Support for New CONCAT_WS Builtin Function

This release of Oracle Rdb adds support for a new builtin function, CONCAT_WS, which is a variation of the CONCAT function. This function uses the first parameter as a separator which is applied after each of the other parameters. For instance, to create a comma separated list of column values, you specify the separator once and have SQL perform the formatting. This is shown in the example below.

```
SQL> select CONCAT_WS (', ', employee_id, birthday, '',
cont>                   last_name, middle_initial, last_name)
cont> from employees
cont> order by employee_id
cont> fetch first 10 rows only;

 00164, 1947-03-28, , Toliver       , A, Toliver
 00165, 1954-05-15, , Smith         , D, Smith
 00166, 1954-03-20, , Dietrich      , Dietrich
 00167, 1937-03-05, , Kilpatrick    , Kilpatrick
 00168, 1932-10-23, , Nash          , Nash
 00169, 1938-08-13, , Gray          , O, Gray
 00170, 1957-06-03, , Wood          , Wood
 00171, 1932-01-29, , D'Amico       , D'Amico
 00172, 1951-05-31, , Peters        , K, Peters
 00173, 1927-03-05, , Bartlett      , G, Bartlett
10 rows selected
SQL>
```

Usage Notes

- If the separator value expression resolves to NULL, then the result of CONCAT_WS will be NULL.
- If any other parameter value expression resolves to NULL, then it will be ignored. That is, that column value and any separator will not be included in the output.
- The function CONCAT_WS accepts all data types with the exception of LIST OF BYTE VARYING, LONG, and LONG RAW. Each non-character string value will be implicitly converted to VARCHAR with a size appropriate for the data type.
  The result of this function will have the type VARCHAR with a length long enough for the concatenated data and separators.
- If dialect ORACLE LEVEL1 or ORACLE LEVEL2 is used, then zero length strings ('') will be considered as NULL and so be excluded from the output. If the resulting value is a zero length string, then the result of CONCAT_WS will be NULL.

Examples

This example shows the use of the CONCAT_WS function to simplify the formatting of table data in CSV (comma separated value) format.

```
SQL> select '"' ||
cont> CONCAT_WS ('", "', first_name, nvl(middle_initial,''), last_name)
cont> || '"'
cont> from employees
cont> order by employee_id;
```

```
 "Alvin     ", "A", "Toliver       "
 "Terry     ", "D", "Smith         "
 "Rick      ", "", "Dietrich      "
 "Janet     ", "", "Kilpatrick    "
...
 "Peter     ", "", "Blount        "
 "Johanna   ", "P", "MacDonald     "
 "James     ", "Q", "Herbener      "
100 rows selected
SQL>
```

1.1.11 New SYSTIMESTAMP Function Added

This release of Oracle Rdb includes a new SYSTIMESTAMP function that returns the current date and time as a TIMESTAMP type. This function is similar to SYSDATE and CURRENT_TIMESTAMP however its type doesn't change when the SET DEFAULT DATE FORMAT command is used.

Syntax

SYSTIMESTAMP [(fractional−seconds−precision)]

Usage Notes

- The function name can be followed by an optional fractional−seconds−precision. This value, if omitted, defaults to 2 and accepts the values 0, 1, or 2.
- The following example shows that SYSTIMESTAMP always returns a SQL standard date and time.

```
SQL> select systimestamp,sysdate,current_timestamp from rdb$database;

 2007−03−27 16:33:32.19   27−MAR−2007 16:33:32.19   27−MAR−2007 16:33:32.19
1 row selected
SQL> set default date format 'sql99';
SQL> select systimestamp,sysdate,current_timestamp from rdb$database;

 2007−03−27 16:33:41.32   2007−03−27 16:33:41.32   2007−03−27 16:33:41.32
1 row selected
SQL>
```

1.1.12 New SET FLAGS Keyword to Control Optimizer Query Rewrite

This release of Oracle Rdb introduces several query rewrite optimizations. These optimizations are enabled by default and can be controlled using the RDMS$SET_FLAGS logical name or the SET FLAGS statement.

The following keywords or keyword parameters can be used. Note that REWRITE, unlike many other SET FLAGS keywords, does not accept a numeric value.

- REWRITE − when no parameters are provided, all query rewrite optimizations are enabled.
- NOREWRITE − when no parameters are provided, all query rewrite optimizations are disabled.
- REWRITE(LIKE), NOREWRITE(LIKE) − specifying the LIKE keyword will enable or disable only the LIKE predicate rewrite.

- REWRITE(STARTING_WITH), NOREWRITE(STARTING_WITH) − specifying the STARTING_WITH keyword will enable or disable only the STARTING WITH predicate rewrite.
- REWRITE(CONTAINING), NOREWRITE(CONTAINING) − specifying the CONTAINING keyword will enable or disable only the CONTAINING predicate rewrite.
- When SET FLAGS 'NONE' or SET NOFLAGS is used, the default setting for REWRITE will be restored.

The following example uses SET and SHOW FLAGS to show the effect of using the NOREWRITE keyword.

```
SQL> set line length 70
SQL> show flags;

Alias RDB$DBHANDLE:
Flags currently set for Oracle Rdb:
   PREFIX,WARN_DDL,INDEX_COLUMN_GROUP,MAX_SOLUTION,MAX_RECURSION(100)
   ,REWRITE(CONTAINING),REWRITE(LIKE),REWRITE(STARTING_WITH)
   ,REFINE_ESTIMATES(127),NOBITMAPPED_SCAN
SQL>
SQL> set flags 'norewrite';
SQL> show flags;

Alias RDB$DBHANDLE:
Flags currently set for Oracle Rdb:
   PREFIX,WARN_DDL,INDEX_COLUMN_GROUP,MAX_SOLUTION,MAX_RECURSION(100)
   ,REFINE_ESTIMATES(127),NOBITMAPPED_SCAN
SQL>
```

1.1.13 New SYS_GUID Function Added

This release of Oracle Rdb supports a new builtin function, SYS_GUID. This function returns a 16 octet globally unique identifier. Applications would use this to provide unique values from various applications and across databases in an OpenVMS cluster or network.

Syntax

SYS_GUID ( )

Usage Notes

- This function uses the OpenVMS system service SYS$CREATE_UID. Applications that call this system service create compatible values for Rdb.
- The returned value from SYS_GUID() may contain octets that are zero. If returning values to C applications, then Oracle recommends using the $SQL_VARCHAR pseudo−type to avoid C null terminated string semantics.
- The SYS_GUID() returns data using a special character set. This special character set is used by Oracle Rdb to distinguish this type of string from others. Interactive SQL will format the value using standard OpenVMS formatting services when this character set is seen. Note that these services perform reordering of the octet values during formatting. That is, the value is not a direct hexadecimal representation of the value.
  Database administrators can define a domain to be used by applications which will make it easier to use.

  ```
  SQL> create domain GUID_DOMAIN
  ```

```
cont>    char(16) character set −11;
SQL>
SQL show domain GUID_DOMAIN;
GUID_DOMAIN                     CHAR(16)
         GUID 16 Characters,  16 Octets
SQL>
```

This domain can be used for column, parameter, and variable definitions.

- To support storing literal GUID values, SQL also supports GUID literals. The literals follow the standard literal format using the special prefix _GUID, as shown in the following examples.

```
SQL> create domain GUID_DOMAIN
cont>    char(16) character set −11;
SQL> show domain GUID_DOMAIN;
GUID_DOMAIN                     CHAR(16)
         GUID 16 Characters,  16 Octets

SQL> create table SAMPLE
cont>     (a int
cont>     ,b GUID_DOMAIN default _guid'00000000−0000−0000−0000−000000000000');
SQL> insert into SAMPLE default values;
1 row inserted

SQL> show table (column) SAMPLE;
Information for table SAMPLE

Columns for table SAMPLE:
Column Name                     Data Type        Domain
−−−−−−−−−−−                     −−−−−−−−−        −−−−−−
A                               INTEGER
B                               CHAR(16)         GUID_DOMAIN
         GUID 16 Characters,  16 Octets
 Oracle Rdb default: GUID'00000000−0000−0000−0000−000000000000'
SQL>
```

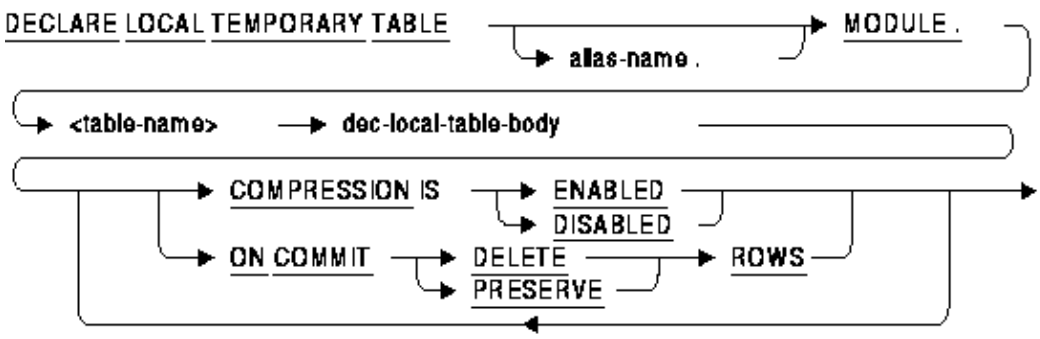
The literal can also be used in queries to select existing rows.

```
SQL> select * from SAMPLE
cont> where b = _guid'3DBB657F−8513−11DF−9B74−0008029189E7';
```

1.1.14 New COMPRESSION Clause for DECLARE LOCAL TEMPORARY TABLE Statement

This release of Oracle Rdb enhances the DECLARE LOCAL TEMPORARY TABLE syntax with a new COMPRESSION option. In prior releases of Rdb, it was not possible to disable row compression for temporary tables even though doing so might save some compression and decompression CPU time overhead. In addition, if the data in the temporary table is not compressible, it is possible that the run−length encoding could increase the resulting row length.

Syntax



Parameters

- COMPRESSION IS ENABLED
  COMPRESSION IS DISABLED
  This clause controls the use of run−length compression for rows inserted into this local temporary table. The default is COMPRESSION IS ENABLED.

Usage Notes

- In some cases, the data inserted into a local temporary table may not compress and so incur only overhead in the row. This overhead is used by Rdb to describe the sequence of uncompressible data. Use COMPRESSION IS DISABLED to prevent Rdb from attempting the compression of such data.

Examples

The following example shows a declared local temporary table that will not benefit from compression. The clause COMPRESSION IS DISABLED is used to reduce the CPU overhead for the table as well as preventing a possible row size increase because of compression notations.

```
SQL> declare local temporary table module.scratch0
cont>     (averages double precision)
cont>     compression is DISABLED
cont>     on commit PRESERVE rows
cont> ;
SQL>
SQL> insert into module.scratch0
cont>     select avg (char_length (a)) from module.scratch1;
1 row inserted
SQL>
SQL> select * from module.scratch0;
                AVERAGES
  2.1000000000000000E+001
```

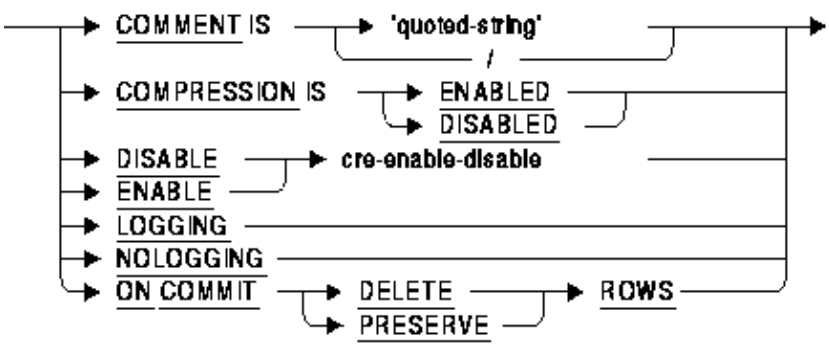
1.1.15 New COMPRESSION Clause for CREATE TABLE Statement

In prior releases of Rdb, it was required that a storage map be created for the table so that row compression

could be disabled. This release of Oracle Rdb enhances the CREATE TABLE syntax with a new COMPRESSION option.

Syntax

create-table-attributes =



Parameters

- COMPRESSION IS ENABLED
  COMPRESSION IS DISABLED
  This clause controls the use of run−length compression for rows inserted into this base or temporary table. The default is COMPRESSION IS ENABLED.

Usage Notes

- In some cases, the data inserted into a table may not compress and so incur only overhead in the row. This overhead is used by Rdb to describe the sequence of uncompressible data. Use COMPRESSION IS DISABLED to prevent Rdb from attempting the compression of such data.
- Any storage map which specifies the ENABLE COMPRESSION or DISABLE COMPRESSION clause will override this setting in the table.
- The COMPRESSION IS clause is not permitted for INFORMATION tables.

Examples

The following example shows that compression was disabled for the created table. The SHOW TABLE statement reports the disabled (that is the non−default) setting for compression.

```
SQL> create table SAMPLE
cont> (ident integer identity
cont> ,sample_value real
cont> )
cont> compression is disabled;
SQL> show table SAMPLE
Information for table SAMPLE

Compression is disabled.
Columns for table SAMPLE:
Column Name                    Data Type        Domain
```

```
-----------                     ---------       ------
IDENT                           INTEGER
 Computed:      IDENTITY
SAMPLE_VALUE                    REAL

Table constraints for SAMPLE:
 No Constraints found

Constraints referencing table SAMPLE:
 No Constraints found

Indexes on table SAMPLE:
 No Indexes found

Storage Map for table SAMPLE:
 No Storage Map found

Triggers on table SAMPLE:
No triggers found

SQL>
```

1.1.16 Support for 2 TiB Storage Area Files

OpenVMS Version 8.4 provides support for disk volumes up to 2 TiB in size. The precise maximum volume size is 4,261,348,350 blocks, which is about 1.98 TiB. Oracle Rdb now supports live and snapshot storage areas with up to 2,147,483,647 database pages and up to the OpenVMS Version 8.4 file size limit of 4,261,348,350 blocks. In order to utilize a database storage area of more than 2,147,483,647 disk blocks (approximately 1 TiB), a database page size of greater than 1 is required; in all cases, a database storage area is limited to 2,147,483,647 pages.

For Oracle Rdb Release 7.2.5, 2 TiB file support is limited to live and snapshot storage areas. Future Oracle Rdb releases are expected to expand this support to cover additional on−disk component files.

Note

The tebibyte is a standards−based binary multiple (prefix tebi, symbol Ti) of the byte, a unit of digital information storage. The tebibyte unit symbol is TiB.

```
1 tebibyte = 2^40 bytes = 1099511627776 bytes = 1024 gibibytes
```

The tebibyte is closely related to the terabyte, which is defined as 10^12 bytes or 1000000000000 bytes, but has been used as a synonym for tebibyte in some contexts.

1.1.17 New RMU/ALTER Feature to Modify the Root and Area Header Unique Identifier

To ensure Oracle Rdb database security and integrity, a Unique Identifier has been added to the database root file and the database storage area file and storage area snapshot file headers. The Unique Identifier in the root

file must match the Unique Identifier in the storage area file headers or a storage area cannot be accessed from the database root. The RMU/ALTER command has been enhanced to allow a Database Administrator to modify and display the database root and storage area Unique Identifier values to prevent problems which will occur if these values are corrupted.

The Unique Identifier values are displayed both in VMS date format surrounded by quotes and as a hexadecimal number surrounded by parentheses. The values displayed are the Unique Identifier values for the current RMU/ALTER session. The Unique Identifier values will not be written to the root or storage area files until the user ends the current session with the RMU/ALTER "COMMIT" command. If the user ends the current session with the RMU/ALTER "ROLLBACK" command, the Unique Identifier values will not be written to the root or storage area files and the Unique Identifier values in effect at the start of the session just ended will be restored for the new session. Any Unique Identifier values that have been changed during the current session will be displayed as "(marked)" before they are committed or rolled back.

The new syntax, which can only be used at the "RdbALTER>" prompt which appears when the RMU/ALTER command is issued at the VMS prompt, is the following where "name" is the storage area name and "id" is the storage area identification number in the database root file. "AREA_HEADER" refers to the storage area file header. "ROOT" refers to the Rdb database root file (*.RDB). "UNIQUE_IDENTIFIER" refers to the Unique Identifier in the storage area header when used with "AREA_HEADER" and to the Unique Identifier in the database root when used with "ROOT". If "SNAPSHOT" is not specified, the storage area (*.RDA) file is assumed. If "SNAPSHOT" is specified, the snapshot storage area (*.SNP) file is assumed. The user cannot specify a new "UNIQUE_IDENTIFIER" value: it must be created by RMU/ALTER.

```
DISPLAY ROOT UNIQUE_IDENTIFIER
```

This command displays the current database root Unique Identifier value.

```
DEPOSIT ROOT UNIQUE_IDENTIFIER (= NEW)
```

If "= NEW" is not specified, this command stores the current database root Unique Identifier value into the storage area header blocks of ALL active storage area and storage area snapshot files which are currently defined in the database root when the user executes the next COMMIT command. If "= NEW" is specified, a new Unique Identifier value is created and stored in both the root file and ALL active storage area file headers when the user executes the next COMMIT command. Note that to ensure database integrity, ALL storage area file headers will be updated. Use the AREA_HEADER commands described below for storing the current root Unique Identifier value in specific designated storage areas.

```
DISPLAY AREA_HEADER {name|id} (SNAPSHOT) UNIQUE_IDENTIFIER
```

This command displays the current storage area file or storage area snapshot file header Unique Identifier value for the storage area with the specified name or number.

```
DEPOSIT AREA_HEADER {name|id} (SNAPSHOT) UNIQUE_IDENTIFIER
```

This command stores the current database root Unique Identifier value into the current storage area file or storage area snapshot file header for the storage area with the specified name or number when the user executes the next COMMIT command.

As stated above, any changes to the area header or root Unique Identifier values will only be written to the actual root and area files when the next "COMMIT" command is executed at the "RdbALTER>" prompt. Any changes to the root or area file headers since the last "COMMIT" command was issued can be undone by

executing the "ROLLBACK" command at the "RdbALTER>" prompt. "COMMIT" and "ROLLBACK" are existing RMU/ALTER commands and affect any current uncomitted changes made in RMU/ALTER, not just changes to the root and storage area header Unique Identifier values.

To execute the DISPLAY or DEPOSIT ROOT or AREA_HEADER command, the user must be attached to the database which the root and areas belong to, either by specifying the database name when issuing the RMU/ALTER command or by executing the "ATTACH" command from the "RdbALTER>" prompt.

The following example shows that for Oracle Rdb single file databases, the Unique Identifier value can only be set for the storage area snapshot file since the storage area is part of the root file. Therefore, the DEPOSIT AREA_HEADER command can only specify the "SNAPSHOT" file or an error will be returned.

```
$ RMU/ALTER PERSONNEL
%RMU-I-ATTACH, now altering database
"DEVICE:[DIRECTORY]PERSONNEL.RDB;1"
  DEPOSIT AREA_HEADER RDB$SYSTEM UNIQUE_IDENTIFIER
%RMU-F-NOTSFDB, This command is not allowed for a single file
 database
  DEPOSIT AREA_HEADER RDB$SYSTEM SNAPSHOT UNIQUE_IDENTIFIER
Area RDB$SYSTEM:
(marked) Root file unique identifier is: "22-OCT-2010 13:49:29.32"
 (00AA557869612643)

 COMMIT
 EXIT
```

Since the DEPOSIT ROOT UNIQUE_IDENTIFIER command always stores the Unique Identifier value in ALL storage area file headers when the user executes the RMU/ALTER "COMMIT" command, it would be redundant to execute the DEPOSIT AREA_HEADER UNIQUE_IDENTIFIER command if a DEPOSIT ROOT UNIQUE_IDENTIFIER command is already pending for the current RMU/ALTER session. Therefore, as the following example shows, in this case a DEPOSIT AREA_HEADER UNIQUE_IDENTIFIER command cannot be executed until the user ends the current session with a COMMIT or ROLLBACK command.

```
$ RMU/ALTER MF_PERSONNEL
%RMU-I-ATTACH, now altering database
 "DEVICE:[DIRECTORY]MF_PERSONNEL.RDB;1"
  DEPOSIT ROOT UNIQUE_IDENTIFIER = NEW
(marked) Root file unique identifier is: "22-OCT-2010 13:49:31.72"
 (00AA55786ACFB115)

  DEPOSIT AREA_HEADER SALARY_HISTORY UNIQUE_IDENTIFIER
%RMU-F-COMROOTCOM, COMMIT or ROLLBACK DEPOSIT ROOT
 UNIQUE_IDENTIFIER command to use this command
  commit
  DEPOSIT AREA_HEADER SALARY_HISTORY UNIQUE_IDENTIFIER
Area SALARY_HISTORY:
(marked) Root file unique identifier is: "22-OCT-2010 13:49:31.72"
 (00AA55786ACFB115)

  COMMIT
  EXIT
```

The following example shows that RMU/ALTER is invoked specifying the database MF_PERSONNEL.RDB. The user then displays the current Unique Identifier value in the database root, creates a new Unique Identifier value in the database root, displays the new Unique Identifier in the root, and

finally specifies "commit" to write the new Unique Identifier value to the database root file and ALL database storage area files. The display messages designate the pending new Unique Identifier value as "(marked)" until the user either executes "commit" to write out the new Unique Identifier value or "rollback" to restore the original Unique Identifier value. The user then verifies the database changes.

```
$ RMU/ALTER MF_PERSONNEL
%RMU-I-ATTACH, now altering database "DISK:[DIRECTORY]MF_PERSONNEL.RDB;1"
  DISPLAY ROOT UNIQUE_IDENTIFIER
         Root file unique identifier is: "22-OCT-2010 13:49:27.87"
 (00AA5578688428BB)
  DEPOSIT ROOT UNIQUE_IDENTIFIER = NEW
(marked) Root file unique identifier is: "22-OCT-2010 13:49:28.34"
 (00AA557868CC9F7A)
  DISPLAY ROOT UNIQUE_IDENTIFIER
(marked) Root file unique identifier is: "22-OCT-2010 13:49:28.34"
 (00AA557868CC9F7A)
  COMMIT
  EXIT
$ RMU/VERIFY/ALL/NOLOG MF_PERSONNEL
```

The following example shows that RMU/ALTER is invoked specifying the database MF_PERSONNEL.RDB. The user then displays the current Unique Identifier value in the root file. He then executes the "deposit" commands to designate that the Unique Identifer value in the root file is to be moved to the DEPARTMENTS area storage and the DEPARTMENTS area snapshot files, displays the Unique Identifier value that is to be moved to the DEPARTMENTS area storage and the DEPARTMENTS area snapshot files, and finally specifies "commit" to actually write the root unique identifier value to the DEPARTMENTS area storage and the DEPARTMENTS area snapshot files. The display messages designate the pending Unique Identifier value as "(marked)" until the user either executes "commit" to write out the Unique Identifier value or "rollback" to restore the original Unique Identifier value. The user then verifies the database changes. The example shows that the user can use either the storage area name or the storage area identifier number in the root to designate the target storage area.

```
$ RMU/ALTER MF_PESONNEL
%RMU-I-ATTACH, now altering database "DEVICE:[DIRECTORY]MF_PERSONNEL.RDB;1"
  DISPLAY ROOT UNIQUE_IDENTIFIER
           Root file unique identifier is: "22-OCT-2010 13:49:28.34"
 (00AA557868CC9F7A)

  DEPOSIT AREA_HEADER DEPARTMENTS UNIQUE_IDENTIFIER
Area DEPARTMENTS:
(marked) Root file unique identifier is: "22-OCT-2010 13:49:28.34"
 (00AA557868CC9F7A)

  DEPOSIT AREA_HEADER 2 SNAPSHOT UNIQUE_IDENTIFIER
Area DEPARTMENTS:
(marked) Root file unique identifier is: "22-OCT-2010 13:49:28.34"
 (00AA557868CC9F7A)

  DISPLAY AREA_HEADER DEPARTMENTS UNIQUE_IDENTIFIER
Area DEPARTMENTS:
(marked) Root file unique identifier is: "22-OCT-2010 13:49:28.34"
 (00AA557868CC9F7A)

  DISPLAY AREA_HEADER 2 SNAPSHOT UNIQUE_IDENTIFIER
Area DEPARTMENTS:
(marked) Root file unique identifier is: "22-OCT-2010 13:49:28.34"
 (00AA557868CC9F7A)
```

```
  COMMIT
  EXIT
$ RMU/VERIFY/ALL/NOLOG MF_PERSONNEL
```

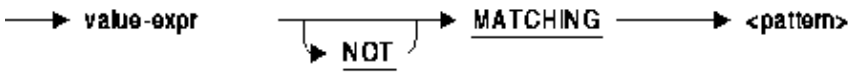
1.1.18 New MATCHING Predicate

This release of Oracle Rdb supports MATCHING as an alternate string pattern matching clause.

Syntax

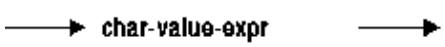
---

matching-predicate =



---

pattern =



A MATCHING predicate searches character string literals for pattern matches. The pattern string accepts the following pattern characters:

- * Matches any string of zero or more characters
- % Matches any single character

Usage Notes

- If either of the expressions is null, the result is null.
- The MATCHING predicate is not case sensitive; it considers uppercase and lowercase forms of the same character to be a match.
- The MATCHING predicate is not sensitive to diacritical markings used in the DEC Multinational Character Set.

The following example shows the use of the MATCHING clause.

```
SQL> select last_name
cont> from employees
cont> where last_name matching '%on*';
 LAST_NAME
 Connolly
 Lonergan
2 rows selected
SQL>
```

1.1.19 New RMU/BACKUP−RESTORE Feature to Check Database Page Integrity

To ensure Oracle Rdb database integrity, a new feature has been added to the RMU/BACKUP and RMU/RESTORE commands to do a basic integrity check of AIP, ABM and data storage area pages when they are backed up and when they are restored. SPAM pages are not backed up by RMU/BACKUP but recreated by RMU/RESTORE based on the page types that are backed up. This basic integrity check will always happen in order to prevent database corruption.

The basic page checks made are not intended to replace an RMU/VERIFY of the database pages before they are backed up or after they are restored. If the page checks report problems, the user should immediately do an RMU/VERIFY of the storage area to get a complete evaluation of the detected problems and any additional problems that may exist. The page checks made are intended to cause minimal additional overhead to the backup or restore operation while indicating serious page corruption exists that needs to be investigated using the RMU/VERIFY and/or RMU/DUMP commands and then corrected. The page checks are necessary so that page integrity problems can be detected and fixed at backup time or so that page integrity problems can be reported and fixed after the restore.

The checks made are a check for a valid storage area id number on the page, a check for a non−zero timestamp on the page, and a check for a page number that is greater than zero and less than or equal to the maximum page number for the storage area. The diagnostic messages output are the same as the diagnostic messages output by RMU/VERIFY if these problems are detected. For RMU/BACKUP, the backup is aborted if any of these checks fail so that the problem can be fixed at backup time. For RMU/RESTORE, the restore will be aborted only if the page number check fails since RMU/RESTORE cannot continue in this case. The other problems will be reported but the backup will continue so the problems can then be fixed in the restored database. Neither the backup or restore operation will be aborted until all the specified checks have been made.

The following example shows that the backup command is aborted because of a page with an invalid page number in storage area SALARY_HISTORY. An RMU/VERIFY of the database gives the additional information of the expected page number where the invalid page number of zero occurs.

```
$ RMU/BACKUP/NOLOG MF_PERSONNEL.RDB MFP.RBF
%RMU−E−BADPAGRAN, area SALARY_HISTORY
%RMU−E−BADPAGRA2,                   page number 0 out of range
%RMU−E−BADPAGRA3,                   expected between 1 and 706
%RMU−F−INVPAGAREA, Aborting command − invalid page 0 in area SALARY_HISTORY
%RMU−F−FATALERR, fatal error on BACKUP
%RMU−F−FTL_BCK, Fatal error for BACKUP operation at 21−DEC−2010
 09:30:19.64
$ rmu/verify/all/nolog mf_personnel
%RMU−W−PAGPAGRAN, area SALARY_HISTORY, page 21
                  page number out of range
                  expected: 21, found: 0
```

The following database backup example shows all three diagnostics that can be put out by the basic page validation checks. If any of these checks fail, the backup operation will be aborted.

```
$ RMU/BACKUP/NOLOG MF_PERSONNEL.RDB MFP.RBF
%RMU−W−PAGBADARE, area RDB$SYSTEM, page 0
%RMU−W−PAGBADAR2,                   maps incorrect storage area
%RMU−W−PAGBADAR3,                   expected: 1, found: 3
%RMU−W−PAGTADZER, area RDB$SYSTEM, page 0
```

```
%RMU-W-PAGTADZE2,                   contains zero time stamp
%RMU-E-BADPAGRAN, area RDB$SYSTEM
%RMU-E-BADPAGRA2,                   page number 0 out of range
%RMU-E-BADPAGRA3,                   expected between 1 and 1012
%RMU-F-INVPAGAREA, Aborting command - invalid page 0 in area RDB$SYSTEM
%RMU-F-FATALERR, fatal error on BACKUP
%RMU-F-FTL_BCK, Fatal error for BACKUP operation at 21-DEC-2010 09:39:21.64
```

The following database restore example shows all three diagnostics that can be put out by the basic page validation checks. The restore operation will be aborted only if the page number is invalid but the other reported problems must be corrected.

```
$ RMU/RESTORE/NOLOG/NOCDD MFP.RBF
%RMU-W-PAGBADARE, area RDB$SYSTEM, page 0
%RMU-W-PAGBADAR2,                   maps incorrect storage area
%RMU-W-PAGBADAR3,                   expected: 1, found: 3
%RMU-W-PAGTADZER, area RDB$SYSTEM, page 0
%RMU-W-PAGTADZE2,                   contains zero time stamp
%RMU-E-BADPAGRAN, area RDB$SYSTEM
%RMU-E-BADPAGRA2,                   page number 0 out of range
%RMU-E-BADPAGRA3,                   expected between 1 and 1012
RMU-F-INVPAGAREA, Aborting command - invalid page 0 in area RDB$SYSTEM
%RMU-F-FATALERR, fatal error on RESTORE
%RMU-F-FTL_RSTR, Fatal error for RESTORE operation at 21-DEC-2010 09:44:01.96
```

1.1.20 New RMU/DUMP/BACKUP /AREA, /START and /END Qualifiers

New /AREA, /START and /END qualifiers have been added to the Oracle Rdb RMU/DUMP/BACKUP command which dumps the contents of an Rdb database backup (*.RBF) file. These qualifiers allow the user to dump backup file records for a specified storage area and for a specified range of pages within the specified storage area.

The syntax for the new qualifiers used with the RMU/DUMP/BACKUP command is the following.

```
/AREA = identity
```

Only dump the storage area identified by the specified name or ID number. The area name must be the name of a storage area in the database root file and the area ID number must be a storage area ID number in the database root file. This information is contained in the "Database Parameters:" section of the backup file which is output at the start of the dump. Snapshot areas are not contained in the backup file and cannot be specified. If this qualifier is used without the new /START and /END qualifiers, all page records in the specified storage area will be output.

```
/START = number
```

Only dump pages starting with the specified page number in the specified storage area. This qualifier cannot be used unless the /AREA qualifier is also specified. If no pages are dumped either the specified page or range of pages does not exist in the specified area in the backup file, or this qualifier has been used in the same RMU/DUMP/BACKUP command as an /OPTIONS, /SKIP or /PROCESS qualifier option that has excluded the specified page or range of pages from the dump. If this qualifier is not used with the new /END qualifier, all page records in the specified storage area starting with the specified page number will be output.

```
/END = number
```

Only dump pages ending with the specified page number in the specified storage area. This qualifier cannot be used unless the /AREA qualifier is also specified. If no pages are dumped either the specified page or range of pages does not exist in the specified area in the backup file, or this qualifier has been used in the same RMU/DUMP/BACKUP command as an /OPTIONS, /SKIP or /PROCESS qualifier option that has excluded the specified page or range of pages from the dump. If this qualifier is not used with the new /START qualifier, all page records in the specified storage area ending with the specified page number will be output.

If both the /START and /END qualifiers are specified, the starting page number must be less than or equal to the ending page number. If the starting page number equals the ending page number only the page records for the specified page number are dumped. The block header for each block which contains at least one of the requested pages is dumped followed by the requested page records in that block. The START AREA record is dumped at the start of requested page records and the END AREA record is dumped at the end of the requested page records. By default, the database root parameters are dumped at the very start following the dump header.

The following example shows the dump of the page records for page 10 in storage area 4 in the MFP.RBF backup file. Since the /START and /END qualifiers both specify page 10, only the page records for that page are dumped. At the start of the dump is the dump header, followed by the database root parameters which are not shown to save space, followed by the block header, which begins with the "HEADER_SIZE" field, for the block which contains the records for page 10 in storage area 4, followed by the start area record for area 4 (REC_TYPE = 6), the data page header record (REC_TYPE = 7) for page 10, the data page data record (REC_TYPE = 8) for page 10, and ending with the end area record for area 4 (REC_TYPE = 11) which ends the dump.

```
$ RMU/DUMP/BACKUP/AREA=4/START=10/END=10/OPTION=FULL MFP.RBF
*-------------------------------------------------------------------------------
* Oracle Rdb V7.2-420                                    11-JAN-2011 15:50:09.25
*
* Dump of Database Backup Header
*     Backup filename: MFP.RBF
*     Backup file database version: 7.2
*
*-------------------------------------------------------------------------------


Database Parameters:
.
.
.

HEADER_SIZE = 80        OS_ID = 1024    UTILITY_ID = 722
APPLICATION_TYPE = 1    SEQUENCE_NUMBER = 22    MAJ_VER = 1    MIN_VER = 1
VOL_NUMBER = 1  BLOCK_SIZE = 32256      CRC = 0C5D3A78  NOCRC = 00
CRC_ALTERNATE = 00      BACKUP_NAME = MFP.RBF   AREA_ID = 4    HIGH_PNO = 259
LOW_PNO = 1     HDR_CHECKSUM = 9B3D

REC_SIZE = 2    REC_TYPE = 6    BADDATA = 00    ROOT = 00
AREA_ID = 4     LAREA_ID = 0    PNO = 0

REC_SIZE = 32   REC_TYPE = 7    BADDATA = 00    ROOT = 00
AREA_ID = 4     LAREA_ID = 0    PNO = 10

REC_SIZE = 28   REC_TYPE = 8    BADDATA = 00    ROOT = 00
AREA_ID = 4     LAREA_ID = 0    PNO = 10
```

```
REC_SIZE = 512  REC_TYPE = 11   BADDATA = 00    ROOT = 00
AREA_ID = 4     LAREA_ID = 0    PNO = 0
```

The following example dumps the records for pages 10, 11 and 12 in the RDB$SYSTEM storage area in the MFP.RBF backup file. Following the block header containing the target records that starts with "HEADER_SIZE =", are the start area record for RDB$SYSTEM area 1 (REC_TYPE = 6), then the target ABM page records for pages 10, 11, and 12 (REC_TYPE = 10), and finally the end area record for area RDB$SYSTEM area 1 (REC_TYPE = 11) which ends the dump.

```
$ RMU/DUMP/BACKUP/AREA=RDB$SYSTEM/START=10/END=12/OPTION=FULL MFP.RBF
*-------------------------------------------------------------------------------
* Oracle Rdb V7.2-420                                    14-JAN-2011 14:40:46.88
*
* Dump of Database Backup Header
*     Backup filename: MFP.RBF
*     Backup file database version: 7.2
*
*-------------------------------------------------------------------------------

Database Parameters:
.
.
.

HEADER_SIZE = 80        OS_ID = 1024    UTILITY_ID = 722
APPLICATION_TYPE = 1    SEQUENCE_NUMBER = 1     MAJ_VER = 1     MIN_VER = 1
VOL_NUMBER = 1  BLOCK_SIZE = 32256      CRC = 8329C24B  NOCRC = 00
CRC_ALTERNATE = 00      BACKUP_NAME = MFP.RBF   AREA_ID = 1     HIGH_PNO = 178
LOW_PNO = 1     HDR_CHECKSUM = 40DE

REC_SIZE = 2    REC_TYPE = 6    BADDATA = 00    ROOT = 00       AREA_ID = 1
LAREA_ID = 0    PNO = 0

REC_SIZE = 10   REC_TYPE = 10   BADDATA = 00    ROOT = 00       AREA_ID = 1
LAREA_ID = 3    PNO = 10

REC_SIZE = 10   REC_TYPE = 10   BADDATA = 00    ROOT = 00       AREA_ID = 1
LAREA_ID = 4    PNO = 11

REC_SIZE = 10   REC_TYPE = 10   BADDATA = 00    ROOT = 00       AREA_ID = 1
LAREA_ID = 4    PNO = 12

REC_SIZE = 512  REC_TYPE = 11   BADDATA = 00    ROOT = 00       AREA_ID = 1
LAREA_ID = 0    PNO = 0
```

1.1.21 Reduced CPU Usage and Improved Performance

Several performance enhancements have been implemented in this release of Oracle Rdb. Most of these changes are either specific to applications running on I64 systems or will have a greater effect on I64 systems. These enhancements include improved code sequences for:

- Integer and floating point arithmetic operations
- Floating point comparison operations
- Floating point conversion operations

1.1.22 New Logical Name to Control Sizing of LIST OF BYTE VARYING Pointer Segments

This release of Oracle Rdb supports a new logical name, RDMS$BIND_SEGMENTED_STRING_PSEG_SIZING, that can be used to control the upper limit for the size of the pointer segment which is stored for LIST OF BYTE VARYING data.

In prior releases, the upper limit for a pointer segment was the free space on a page. If only a small number of LIST segments were inserted, then the pointer segment was trimmed to that required by the data for that column. However, when many relatively small segments were stored there would be one or more large segments stored that required Rdb to search for free space (essentially a free storage area page). The result was a high number of discarded pages − pages read from disk that contained free space but insufficient free space for the stored pointer segments. The solution is to define THRESHOLD values for the MIXED format storage area, or the LIST storage map for UNIFORM format storage areas so that Rdb has guidance on acceptable pages.

To simplify the management of LIST OF BYTE VARYING data, Rdb now supports this new logical to control INSERT behavior for pointer segments. The logical name RDMS$BIND_SEGMENTED_STRING_PSEG_SIZING can be defined to one of the following numeric values:

- 0
  Use the current page size as the upper limit. This remains the default behavior (as in previous releases) if this logical name is not defined.
- 1
  Use the maximum segment length for the current LIST OF BYTE VARYING column value. For instance, if the application always stored 100 octet values in the LIST then this will be used (plus record overhead) to limit the size of the pointer segments.
- 2
  Use the average segment length for the current LIST OF BYTE VARYING column value. This is a useful setting when segments are of different sizes, such as lines from a text document.

Usage Notes

- If any other value is used for the logical name, then it will be ignored and the setting will default to standard behavior.
- This logical name affects all tables including system tables.
- The effect may be more smaller pointer segments. This will translate to increased I/O counts. However, the benefit should be better placement in the storage area and elimination (or reduction) in discarded pages.
- If the average or maximum length of the data segments is too small, then Rdb will ensure that the pointer segments can store at least three pointers. When stored, the pointer segments will be trimmed to contain only valid data pointers.
- The stored data is compatible with all releases of Oracle Rdb, and the logical can be deassigned or redefined at any time.
- If the logical name RDMS$USE_OLD_SEGMENTED_STRING is defined as "T", "t", or "1" then Rdb will revert to chained style segmented strings. In that case, the value specified by RDMS$BIND_SEGMENTED_STRING_PSEG_SIZING will not be used.

1.1.23 RMU /BACKUP Performance Improvements

RMU /BACKUP performance has been improved by streamlining code sequences and reducing redundant data copies in memory. In some cases, reductions of up to 10% of CPU time have been realized.

1.1.24 New RMU/BACKUP/ENCRYPT "%RMU−I−ENCRYPTUSED" Message Added

The Oracle Rdb RMU/BACKUP/ENCRYPT and RMU/BACKUP/AFTER_JOURNAL/ENCRYPT commands create encrypted Rdb database backup files and Rdb database After Image Journal backup files using an encryption key. The same encryption key must be specified when the database is restored or recovered by the RMU/RESTORE/ENCRYPT and RMU/RECOVER/ENCRYPT commands. The following informational message will now always be output on a successful completion of the RMU/BACKUP/ENCRYPT and RMU/BACKUP/AFTER_JOURNAL/ENCRYPT commands to remind the user that the same key specified for these commands must be specified when the created backup files are restored or recovered by the RMU/RESTORE/ENCRYPT and "RMU/RECOVER/ENCRYPT commands.

```
%RMU−W−ENCRYPTUSED, Encryption key required when future restore
performed.
```

The following example shows this new informational message being put out when successful RMU/BACKUP/ENCRYPT and RMU/BACKUP/AFTER_JOURNAL/ENCRYPT commands are completed.

```
$ RMU/BACKUP/NOLOG database_file backup_file −
 /ENCRYPT=(VALUE=key_name,ALGORITHM=algorithm_name)
%RMU−I−ENCRYPTUSED, Encryption key required when future restore
performed.
$ RMU/BACKUP/AFTER_JOURNAL/FORMAT=NEW_TAPE −
 database_file backup_file −
 /ENCRYPT=(VALUE=key_name,ALGORITHM=algorithm_name)/NOLOG
%RMU−I−ENCRYPTUSED, Encryption key required when future restore
performed.
```

1.1.25 New DATABASE_HANDLE Option for the GET DIAGNOSTICS Statement

This release of Oracle Rdb adds the keyword DATABASE_HANDLE for use by the GET DIAGNOSTICS statement. This option returns a unique handle (or stream) identifier which can be useful in distinguishing one attach from another.

```
SQL> set flags 'trace';
SQL>
SQL> begin
cont> declare :ii integer;
cont> get diagnostics :ii = DATABASE_HANDLE;
cont> trace :ii;
cont> end;
~Xt: 1
SQL>
```

1.1.26 New SYS_GET_DIAGNOSTIC Function Supported for SQL

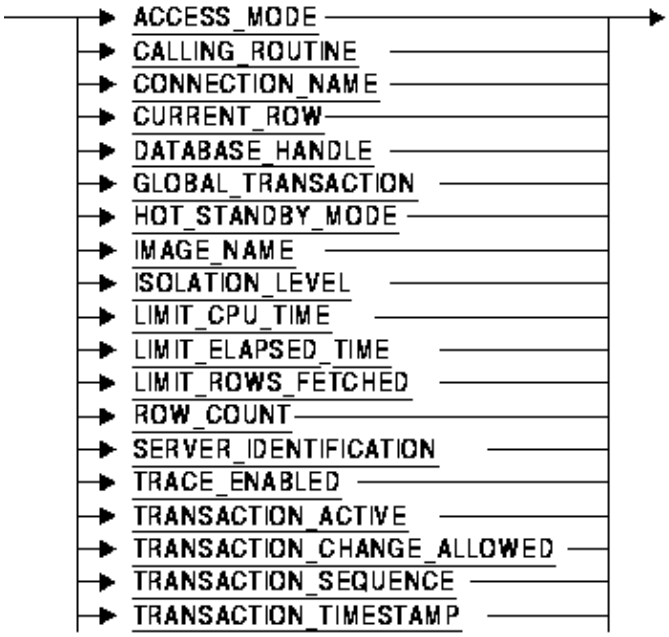
This release of Oracle Rdb adds a new function, SYS_GET_DIAGNOSTIC, that can be used to return the same session information available to the GET DIAGNOSTICS statement. This function provides a shorthand method of fetching values without requiring a compound statement or an intermediate variable.

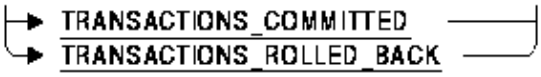
Syntax

SYS_GET_DIAGNOSTIC (statement−item−name) −−>

---

statement-item-name =





Usage Notes

- For the list of keywords acceptable by this function, please see the statement−item−name syntax under the GET DIAGNOSTICS statement.
- Each keyword used with SYS_GET_DIAGNOSTIC will cause the result to have a different associated data type. Please refer to the GET DIAGNOSTICS statement for the returned data types.

Examples

The following example shows the return of session information.

```
SQL> select SYS_GET_DIAGNOSTIC (CONNECTION_NAME) as CONN,
cont>       SYS_GET_DIAGNOSTIC (SERVER_IDENTIFICATION) as IDENT,
cont>       SYS_GET_DIAGNOSTIC (DATABASE_HANDLE) as DBHANDLE
cont>  from GET_DIAG;
 CONN                              IDENT                               DBHANDLE
 RDB$DEFAULT_CONNECTION            Oracle Rdb V7.2−501                    1
1 row selected
SQL>
```

1.1.27 Improved Error Handling for Database Disk Backup File Sets

Bug 4596098

The "/DISK_FILE" command qualifier can be used with certain Oracle Rdb RMU commands such as "RMU/BACKUP" and "RMU/RESTORE" to create or read database disk backup file sets. The error handling for the "/DISK_FILE" qualifier used with the "RMU/RESTORE" and "RMU/DUMP/BACKUP" commands has been improved in the following cases.

If the first file of a backup file set is not specified, the "RMU/RESTORE" or "RMU/DUMP/BACKUP" command cannot continue since the first file of the backup file set contains the backed up root. For this case, a new message has been added which displays the backup command that was used to create the backup file set which is contained in the summary record at the start of each backup file. The backup command will show all the backup files created by the backup in the correct order.

```
%RMU-W-BCKCMDUSED, The backup command was
"RMU/BACKUP...".
```

In addition, a new message has been added to specify that file number "n" in the backup file set, the first backup set file specified in the restore command, is not the first backup file of the backup set created by the backup command.

```
%RMU-E-NOTFIRVOL, Backup set file number n, the first backup file specified,
is not the first file of the backup set. Specify all the backup set files or
devices in the correct order.
```

The following example shows the new error messages for this case. The first backup set file created by the "RMU/BACKUP/DISK_FILE" command, "MFP", is not specified by the "RMU/RESTORE" command.

```
$ RMU/RESTORE/NOLOG/NOCDD/DISK=READER=3 MFP01,MFP02,MFP03
%RMU-W-BCKCMDUSED, The backup command was
"RMU/BACKUP/NOLOG/DISK=WRITER=4 MF_PERSONNEL MFP,MFP01,MFP02,MFP03".
%RMU-E-NOTFIRVOL, Backup set file number 2, the first backup file specified,
is not the first file of the backup set. Specify all the backup set files or
devices in the correct order.
%RMU-F-INVDBBFIL, invalid backup file DEVICE:[DIRECTORY]MFP01.RBF;
%RMU-F-FATALERR, fatal error on RESTORE
%RMU-F-FTL_RSTR, Fatal error for RESTORE operation at 22-MAR-2011 10:04:26.82
```

If the first file of the backup file set is specified but other file(s) of the set are left out, a new message has been added which displays the backup command that was used to create the backup file set which is contained in the summary record at the start of each backup file. This will show all the backup files created by the backup command in the correct order.

```
%RMU-W-BCKCMDUSED, The backup command was
"RMU/BACKUP...".
```

Also, the warning message below has been changed to the following error message.

```
%RMU-W-RESINCOMP, Not all storage areas have been restored
```

```
%RMU-E-RESINCOMP, Not all storage areas have been restored, the
 database may be corrupt.
```

In this case, "%RMU−E−RESINCOMP" has been made the exit status. The restored database can be attached to and the storage areas that were restored can be accessed in SQL without error, though SQL will return an

error if access is attempted to the storage areas in the root that were not restored. The user should delete the restored files and repeat the restore using all the backup files in the backup set displayed in the "%RMU−W−BCKCMDUSED" message. If any of the backup files in the set have been lost, the missing storage areas can be restored by the RMU/RESTORE/AREA command using a previous backup file if it is available.

The new error messages for this case are not put out for the "RMU/DUMP/BACKUP" command since this command can be used to dump one or more files of a backup set without error as long as the first backup set file containing the root information is included as the first file or only file specified. If the "/OPTION=DEBUG" qualifier is used with the "RMU/DUMP/BACKUP" command, the backup command will be displayed along with the other summary record fields at the start of each backup set file.

The following example shows the new error messages for this case. The second backup set file created by the "RMU/BACKUP/DISK_FILE" command, "MFP01", is not specified by the "RMU/RESTORE" command.

```
$ RMU/RESTORE/NOLOG/NOCDD/DISK=READER=3 MFP,MFP02,MFP03
%RMU−I−AIJRSTAVL, 0 after−image journals available for use
%RMU−I−AIJISOFF, after−image journaling has been disabled
%RMU−W−USERECCOM, Use the RMU Recover command. The journals are not available.
%RMU−W−BCKCMDUSED, The backup command was
"RMU/BACKUP/NOLOG/DISK=WRITER=4 MF_PERSONNEL MFP,MFP01,MFP02,MFP03".
%RMU−E−RESINCOMP, Not all storage areas have been restored, the database may be
corrupt.
```

If, for the command "RMU/RESTORE/DISK_FILE/READER_THREADS=number", the number of reader threads specified was greater than the number of backup files specified, the following fatal message was put out and the restore was terminated. This message made sense only for tape backups where you can specify the "/VOLUMES" and "/MASTER" qualifiers.

```
%RMU−F−CONFLSWIT, conflicting qualifiers /VOLUMES and /MASTER
```

The %RMU−F−CONFLSWIT message has been replaced by the more general fatal error message below.

```
%RMU−F−EXTRAREADERS, "n", the number of reader threads, exceeds "n",
 the number of output files or master tape devices.
```

This message will not be displayed for the "RMU/DUMP/BACKUP" command which does not allow the number of reader threads to be specified.

The following example shows this last case. Five reader threads are specified for the "RMU/RESTORE/DISK_FILE" command but there are only four backup files in the backup file set so the first restore command fails. If four or fewer reader threads are specified, the command will succeed.

```
$ RMU/RESTORE/NOLOG/NOCDD/DISK=READER=5 MFP,MFP01,MFP02,MFP03
%RMU−F−EXTRAREADERS, "5", the number of reader threads, exceeds "4", the number
of output files or master tape devices.
%RMU−F−FTL_RSTR, Fatal error for RESTORE operation at 22−MAR−2011 10:17:08.23
$ RMU/RESTORE/NOLOG/NOCDD/DISK=READER=4 MFP,MFP01,MFP02,MFP03
%RMU−I−AIJRSTAVL, 0 after−image journals available for use
%RMU−I−AIJISOFF, after−image journaling has been disabled
%RMU−W−USERECCOM, Use the RMU Recover command. The journals are not available.
$ RMU/VERIFY/ALL MF_PERSONNEL
```

Chapter 2
Enhancements And Changes Provided in Oracle Rdb Release 7.2.4.2

2.1 Enhancements And Changes Provided in Oracle Rdb Release 7.2.4.2

2.1.1 Intel Itanium Processor 9300 "Tukwila" Support

For this release of Oracle Rdb on HP Integrity servers, the Intel Itanium Processor 9300 series, code named "Tukwila", is the newest processor supported.

Chapter 3
Enhancements And Changes Provided in Oracle Rdb Release 7.2.4.1

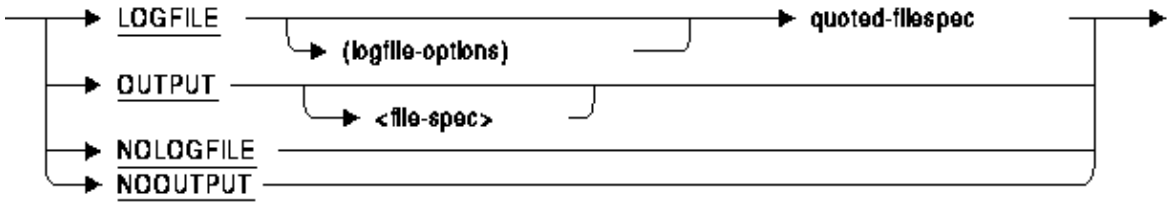
3.1 Enhancements And Changes Provided in Oracle Rdb Release 7.2.4.1

3.1.1 New SET LOGFILE Command

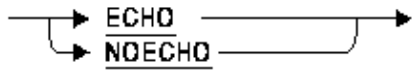
This release of Oracle Rdb adds a new SET LOGFILE statement to interactive SQL. This statement allows the executing SQL script to save output to an OpenVMS file.

Syntax

set-output=



logfile-options =



Arguments

- quoted−filespec
  A valid OpenVMS file specification. Output from interactive SQL will be written to this file.
- NOLOGFILE
  Closes the current output file specified by a prior SET LOGFILE (or SET OUTPUT command).
- NOOUTPUT
  Suspends writing to the output file.
- ECHO
  In addition to writing the output to the designated file, all commands and errors generated by interactive SQL are also written to SYS$OUTPUT.
- NOECHO
  Disable output to SYS$OUTPUT. All commands and errors generated by interactive SQL are only written to the output file.

Usage Notes

- SET LOGFILE is functionally equivalent to the SET OUTPUT statement. However, the SET OUTPUT statement is limited in functionality and is maintained for backward compatibility.

- Files opened with SET OUTPUT and SET LOGFILE can be subsequently processed by either a SET NOOUTPUT or a SET NOLOGFILE command.
- A SET LOGFILE command that does not specify a file is equivalent to SET NOLOGFILE.
- Output written by external functions, SQL TRACE statements, and other output enabled by the SET FLAGS command is never written to the SQL log file. Therefore, it cannot be captured using the statement.

Examples

Saving the output from a script

The following example shows the use of SET LOGFILE to save the output from a script without echoing the results.

1. The script being executed.

```
set verify;
start transaction read only;
set logfile (noecho) 'saved_date.log';
select rdb$flags from rdb$database;
set nologfile;
show alias;
rollback;
```

2. The output as seen during the Interactive SQL session.

```
SQL> start transaction read only;
SQL>
SQL> set logfile (noecho) 'saved_date.log';
SQL>
SQL> show alias;
Default alias:
    Oracle Rdb database in file SQL$DATABASE
SQL> rollback;
```

3. The output saved in the log file.

```
SQL>
SQL> select rdb$flags from rdb$database;
   RDB$FLAGS
           0
1 row selected
SQL>
SQL> set nologfile;
```

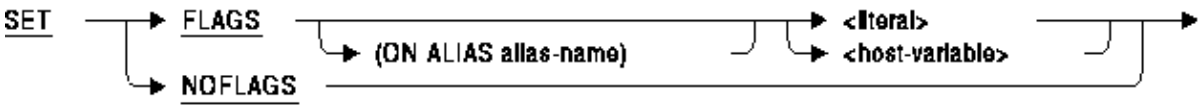
3.1.2 SET FLAGS Statement Now Allows ON ALIAS Clause

This release of Oracle Rdb extends the SET FLAGS statement to support an ON ALIAS clause. The default behavior for SET FLAGS is to establish the flag settings on all currently attached databases. This new clause will allow the database administrator to set flags on just one database alias.

The following example shows a case where enabling AUTO_OVERRIDE required DBADM privilege on the target database but not on the source database. It may be that the current user does not have (or really need) DBADM privilege on that database.

```
SQL> -- Now enable AUTO_OVERRIDE on only one database
SQL> set flags (on alias abc_a) 'auto_override';
SQL> set flags (on alias abc_b) 'none';
SQL> insert into abc_a.SAMPLE_TABLE select * from abc_b.SAMPLE_SOURCE;
SQL> commit;
```

Syntax



Arguments

- alias−name
  The name of an alias as declared by the ATTACH or CONNECT statement. If no ALIAS clause is used, then the alias name will default to RDB$DBHANDLE.

3.1.3 SQL Compiler−Generated Name Uniqueness Enhanced

Bug 4119771

In previous versions of Oracle Rdb, the SQL module language compiler (SQLMOD) and the SQL precompiler (SQLPRE) would generate object names based solely on the system time. This could, in some cases, result in duplicate names being generated for multiple different objects that were compiled at the same time by different processes.

This problem, for example, could cause linker warnings that show the symbol with a name similar in format to "SQL$PROC_1_A3DB62_3331BC" that had been created twice from within different object modules.

This problem has been corrected in Oracle Rdb Release 7.2.4.1. The SQL module language compiler and the SQL precompiler now create unique names within a system or a cluster. The names are comprised of a prefix, a request number and a string comprised of a cluster−wide unique value. The unique value includes components of the system time and the ID of the compiling process. The format of the new names is similar to "SQL$PRC9_DJHS2IHDBAA1G8BQ26A0".

3.1.4 Reduced CPU Usage and Improved Performance

Several performance enhancements have been implemented in this release of Oracle Rdb. Most of these changes are either specific to applications running on I64 systems or will have a greater effect on I64 systems. These enhancements include:

- Streamlined code sequences
- Reduced alignment faults

3.1.5 New RMU /SHOW STATISTICS /WRITE_REPORT_DELAY=n Feature

Bug 3199615

Previously, it was not possible to use the RMU /SHOW STATISTICS to write a report file in non−interactive mode.

This problem has been corrected in Oracle Rdb Release 7.2.4.1. The /WRITE_REPORT_DELAY=n qualifier specifies that statistics are to be collected for "n" seconds (default of 60 seconds) and then a report file written and then the RMU /SHOW STATISTICS utility will exit. /WRITE_REPORT_DELAY implies /NOINTERACTIVE.

Chapter 4
Enhancements And Changes Provided in Oracle Rdb Release 7.2.4

4.1 Enhancements And Changes Provided in Oracle Rdb Release 7.2.4

4.1.1 Date/Time Arithmetic Enhancements

Bug 6219485

This release of Oracle Rdb lifts many restrictions on the DATE VMS data type. SQL now treats DATE VMS type as a TIMESTAMP(2) for the purposes of the add and subtract with intervals or when generating intervals.

- A year−month interval can be added to or subtracted from a DATE VMS value and result in a DATE VMS value.
- A day−time interval can be added to or subtracted from a DATE VMS value and result in a DATE VMS value.
- A DATE VMS value can be subtracted from another DATE VMS value to produce either a year−month interval or a day−time interval.
- A DATE ANSI value can be subtracted from a DATE VMS value to produce a year−month interval.
- A DATE VMS value can be subtracted from a DATE ANSI value to produce either a year−month interval or a day−time interval.
- A DATE VMS value can be subtracted from a TIMESTAMP value to produce either a year−month interval or a day−time interval.
- A TIMESTAMP value can be subtracted from a DATE VMS value to produce either a year−month interval or a day−time interval.

Also in this release the following rules have been relaxed.

- Relax rule that TIME values could only be subtracted if they had the same fractional seconds precision.
- Relax rules that TIMESTAMP values could only be subtracted if they had the same fractional seconds precision.
- Relax assignment rules and let Rdb handle truncating time portion when TIMESTAMP is assigned to a DATE ANSI column, variable or parameter.
- Add rule that merge of DATE VMS and TIMESTAMP(n) will always be TIMESTAMP(n), where n is inherited from the TIEMSTAMP expression. Merge rules are used by UNION, INTERSECT, EXCEPT, MINUS operators and CASE expression processing.

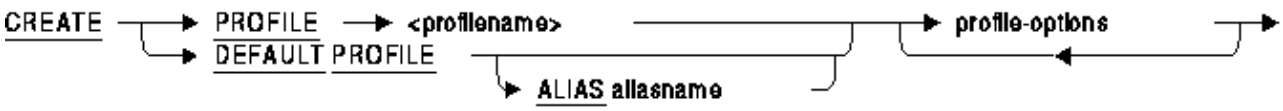
These enhancements have been made in Oracle Rdb Release 7.2.4.

4.1.2 New DEFAULT PROFILE Feature

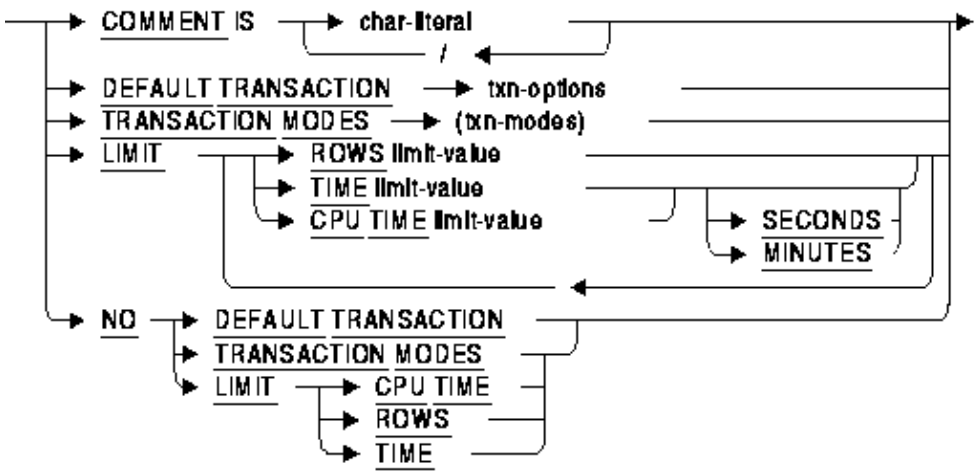
This release of Oracle Rdb enhances the PROFILE support with a new DEFAULT profile. When a user attaches to the database using ATTACH, CONNECT or SET SESSION AUTHORIZATION, they will either load their assigned profile definition or inherit the default profile (if defined).

Syntax

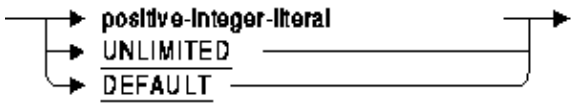
The CREATE, ALTER and DROP PROFILE syntax is changed as shown. The existing profile−options diagram remains the same.

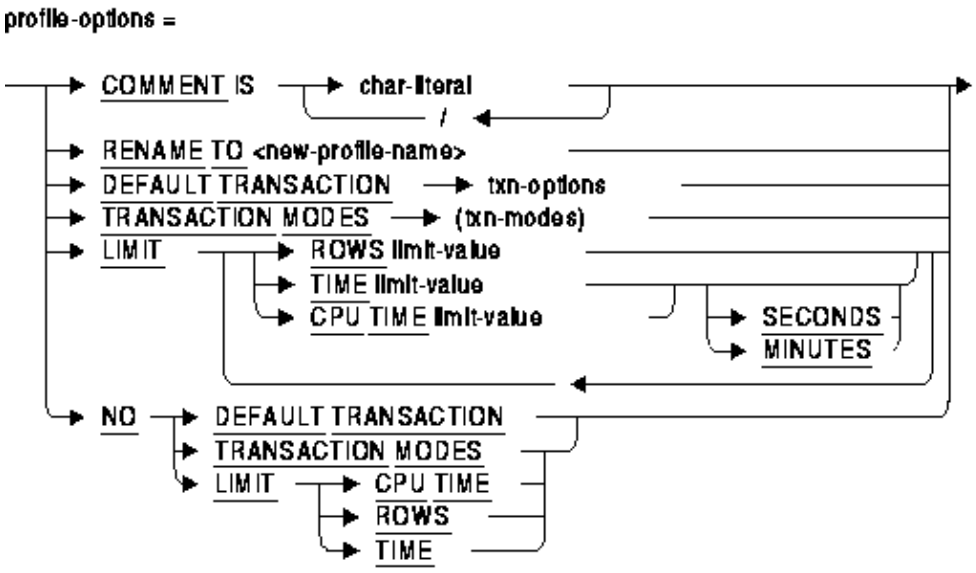


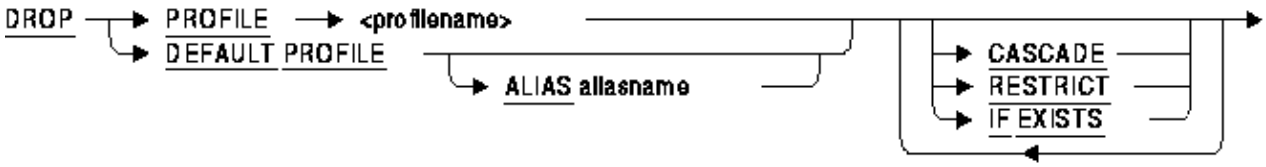
profile-options =



limit-value =







Arguments

- ALIAS aliasname
  When attached to multiple databases, the aliasname is required to direct the CREATE, ALTER or DROP command to the appropriate database.
- DEFAULT PROFILE
  Creates the special profile RDB$DEFAULT_PROFILE. This profile will be used by any user who is not assigned a profile using the PROFILE clause of CREATE or ALTER PROFILE.

Usage Notes

- It is possible to restrict the transaction modes to READ ONLY using the default profile. Use caution in this case because it is possible that no user will have READ WRITE access to undo such a definition. In this case, you can define the logical name RDMS$SET_FLAGS to the value PROFILE_OVERRIDE to allow a suitably privileged user to start a transaction without using the transaction mode restrictions in the default profile. Such a user must have database SECURITY privilege, possibly inherited from the OpenVMS SECURITY process privilege.

4.1.3 RMU /DUMP/BACKUP/OPTIONS=ROOT /HEADER_ONLY Displays the Header Information Only

Bug 8235615

A new feature has been added to RMU/DUMP/BACKUP/OPTIONS=ROOT command to process only the header information when the /HEADER_ONLY qualifier is used.

In prior releases of Oracle Rdb, the user had to wait until the entire backup file (.RBF) was processed. If the backup file was stored on tape and spanned multiple tapes then all the tapes had to be mounted and processed. When using /HEADER_ONLY, RMU now ceases processing of the backup file once the header has been displayed.

```
$ RMU/DUMP /BACKUP/OPTIONS=ROOT /HEADER_ONLY DBNODE$LMA200:GLORY.RBF
*-------------------------------------------------------------------------------
* Oracle Rdb V7.2-4                                        26-FEB-2009 07:21:58.69
*
* Dump of Database Backup Header
*     Backup filename: GLORY.RBF
*     Backup file database version: 7.2
*
*-------------------------------------------------------------------------------

Database Parameters:
    Root filename is "USER1:[BUG.8235615.FIX]GLORY.RDB;1"
    Created at 25-APR-2007 16:43:05.52
    Oracle Rdb structure level is 72.1
    Maximum user count is 23
    Maximum node count is 3
    Database open mode is AUTOMATIC
    Database close mode is AUTOMATIC
    Database will be mapped in process space
    All transaction modes are allowed
    Prestarted transactions are enabled
    Snapshot mode is NON-DEFERRED
    Statistics are enabled
    Operator notification is disabled
    Logical area count is 512
    Storage Areas...
      - Active storage area count is 4
      - Reserved storage area count is 8
    Row Caches...
      - Active row cache count is 0
      - Reserved row cache count is 1
      - Checkpoint information
          No time interval is specified
          Default source is updated rows
          Default target is backing file
          Default backing file directory is database directory
          RUJ Global Buffers are disabled
      - WARNING: Maximum node count is 3 instead of 1
      - WARNING: After-image journaling is disabled
      - WARNING: Fast commit is disabled
    Buffers...
      - Default user buffer count is 2000
      - Default recovery buffer count is 2000 (stored as 20)
      - Global buffers are disabled
          Global buffer count is 115
```

```
      Maximum global buffer count per user is 5
      Large memory is disabled
  - Buffer size is 12 blocks
      Maximum pages per buffer is 6
  - Asynchronous pre-fetch is enabled
      Maximum pre-fetch depth is 8 buffers
  - Detected asynchronous pre-fetch is enabled
      Maximum pre-fetch depth is 4 buffers
      Pre-fetch threshold is 4 buffers
  - Asynchronous batch-write is enabled
      Clean buffer count is 5
      Maximum batch size is 10 buffers
  - Optimized page transfer is disabled
Locking...
  - Adjustable record locking is enabled
      Fanout factor 1 is 10 (10 pages)
      Fanout factor 2 is 10 (100 pages)
      Fanout factor 3 is 10 (1000 pages)
  - Carry-over lock optimization is enabled
  - Lock tree partitioning is disabled
RUJ Journaling...
  - No default recovery-unit journal directory
AIJ Journaling...
  - After-image journaling is disabled
  - Database is configured for 7 journals
  - Reserved journal count is 7
  - Available journal count is 0
  - LogMiner is disabled
  - 7 journals can be created while database is active
  - Shutdown time is 60 minutes
  - Backup operation is manual
  - Default backup filename edits are not used
  - Log server startup is MANUAL
  - Journal overwrite is disabled
  - AIJ cache on "electronic disk" is disabled
  - Default journal allocation is 512 blocks
  - Default journal extension is 512 blocks
  - Default journal initialization is 512 blocks
  - Current roll-forward sequence number is 0
  - Current backup sequence number is 0
Fast Commit...
  - Fast commit is disabled
  - No checkpointing AIJ interval is specified
  - No checkpointing time interval is specified
  - No checkpointing transaction interval is specified
  - Commit to AIJ optimization is disabled
  - Transaction interval is 256
Hot Standby...
  - WARNING: After-image journaling is disabled
  - WARNING: Fast commit is disabled
  - WARNING: Log server startup is MANUAL
  - Informational: Operator notification is disabled
  - Database is not currently being replicated
Security Auditing...
  - Security auditing is disabled
  - Security alarm is disabled
  - No audit journal filename is specified
  - No alarm name is specified
  - Synchronous audit record flushing is disabled
  - Audit every access
Database Backup...
  - Fast incremental backup is enabled
```

```
      - Last full database backup was on 26-FEB-2009 07:16:56.09
      - Full database backup TSN is 0:128
      - Database was restored on 26-FEB-2009 06:52:11.82
    Derived Data...
      - Global section size
          With global buffers disabled is 277187 bytes (1MB)
          With global buffers enabled is 1036584 bytes (1MB)
            With Large memory global buffers enabled...
              Database TROOT section is 330024 bytes (1MB)
              Large memory global buffers section is 706560 bytes (1MB)
      - Row Cache RUJ buffers section size is 6041088 bytes (6MB)

Database root file ACL

(IDENTIFIER=[RDB,SFRANN],ACCESS=READ+WRITE+CONTROL+RMU$ALTER+RMU$ANALYZE+
     RMU$BACKUP+RMU$CONVERT+RMU$COPY+RMU$DUMP+RMU$LOAD+
     RMU$MOVE+RMU$OPEN+RMU$RESTORE+RMU$SECURITY+RMU$SHOW+RMU$UNLOAD+RMU$VERIFY)
```

4.1.4 GET ENVIRONMENT Now Supports SQLCODE and SQLSTATE Capture

This release of Oracle Rdb allows the GET ENVIRONMENT (SESSION) command to return the SQLCODE and SQLSTATE of the last executed statement. The keyword SQLCODE returns an INTEGER value and SQLSTATE returns a CHAR(5) value. The execution of the GET ENVIRONMENT statement will clear these values so both should be fetched in the same statement.

The following example shows a raised error and the use of GET ENVIRONMENT to capture the SQLCODE and SQLSTATE.

```
SQL> declare :st char(5);
SQL> declare :sc integer = -1;
SQL>
SQL> begin set :sc = :sc / 0; end;
%RDB-E-ARITH_EXCEPT, truncation of a numeric value at runtime
-COSI-F-ARITH, arithmetic exception
-COSI-F-FLTDIV, floating point division exception
SQL>
SQL> get environment (session) :st = SQLSTATE, :sc = SQLCODE;
SQL>
SQL> print :st, :sc;
 ST               SC
 22003          -304
SQL>
```

Once the SQLCODE or SQLSTATE value has been saved to a declared variable, it can be used to conditionally execute a COMMIT or a ROLLBACK.

```
begin
if :sc < 0 then
    rollback;
else
    commit;
end if;
end;
```

4.1.5 Timestamp Added to Messages For RMU LOAD and UNLOAD

In order to help judge progress of RMU LOAD and UNLOAD operations, a timestamp has been added to the RMU−I−DATRECSTO and RMU−I−DATRECUNL progress messages. The following example shows these timestamps.

```
$RMU /UNLOAD MFP C1 C1
%RMU-I-DATRECUNL,   200000 data records unloaded  5-JUN-2009 07:58:17.23.
$RMU /LOAD /COMMIT=75000 /LOG_COMMIT MFP C1 C1
%RMU-I-DATRECSTO,   75000 data records stored  5-JUN-2009 07:58:43.09.
%RMU-I-DATRECSTO,   150000 data records stored  5-JUN-2009 07:58:43.12.
%RMU-I-DATRECSTO,   200000 data records stored  5-JUN-2009 07:58:43.14.
```

4.1.6 New SET SQLDA Statement

Bugs 1088554, 4179408, 5414051, and 7022262

This release of Oracle Rdb introduces a new SET SQLDA statement.

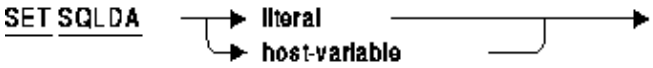
The SQLDA Statement allows a programmer using Dynamic SQL to alter the way the SQLDA (and SQLDA2) and Dynamic SQL statements are processed by Oracle Rdb.

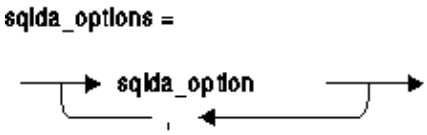
Environment

You can use the SET SQLDA statement:

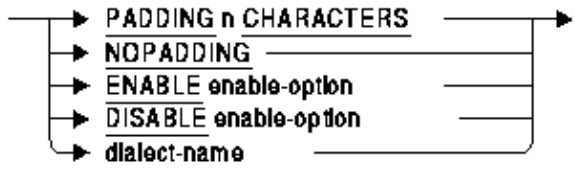
- In dynamic SQL as a statement to be dynamically executed

Syntax

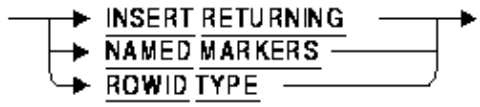




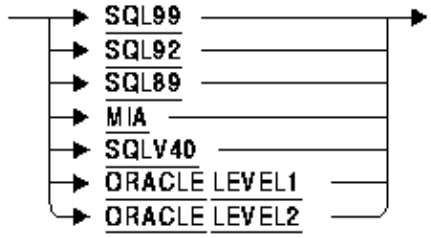
sqlda_option =



enable-option =



dialect-name =



Arguments

- literal
  host−variable
  The parameter passed to the statement must be a literal or a host variable containing one or more SQLDA options (see sqlda_options syntax diagram for details). If more than one option is specified, they must be separated by commas.
- sqlda_options
  One or more keyword clauses. If more than one clause is specified, they must be separated by commas.
- ENABLE
  The ENABLE clause activates one of the following behaviors for Dynamic SQL.
  - ♦ INSERT RETURNING − The default behavior of INSERT ... RETURNING when executed by dynamic SQL is to place parameters from the RETURNING INTO clause into the INPUT SQLDA. This behavior is maintained for backward compatibility. This option allows the programmer to force different (and correct) behavior for the non−compound use of this statement.

Note

If the INSERT RETURNING statement is included in a compound statement, the parameters are handled correctly.

- ♦ NAMED MARKERS − as well as traditional parameters markers (?). Dynamic SQL will now accept named, host variable style parameter markers. See the Usage Notes for further details and examples.
- ♦ ROWID TYPE − returns DBKEY values as a special type (SQLDA_ROWID, 455) to make processing of the DBKEY values easier. For instance, in prior releases, the SQLDA name field (SQLNAME) for DBKEY entries in the SQLDA was the only way to distinguish these values from other CHAR or VARCHAR columns − it would be either DBKEY or ROWID. If a query renamed the DBKEY column, then the application had no information in the SQLDA to indicate that the CHAR or VARCHAR value was binary data. In all respects, the SQLDA_ROWID type appears as a fixed length string of octets (possibly containing octets of zero which the C language would treat as a NULL terminator for a string).

- DISABLE
  The DISABLE clause deactivates one of the specified behaviors for Dynamic SQL. See ENABLE clause for a list of options.
- ORACLE LEVEL1
  ORACLE LEVEL2
  Either of these options will set the SQLDA to supply enhanced semantics. These options are currently reserved for use of the OCI Services for Rdb product that is part of the Oracle Rdb SQL/Services component. This setting also implicitly enables NAMED MARKERS.
- PADDING n CHARACTERS
  This option directs SQL to configure the SQLDA with larger CHARACTER VARYING strings than would normally be seen. The value of n is an unsigned numeric literal that specifies the number of characters that are added to the estimated length. Any CHARACTER (CHAR) types are converted to CHARACTER VARYING (VARCHAR). This rule is applied to comparison operators <, <=, >, >=, =, <>, and string functions (STARTING WITH, CONTAINING).
- NOPADDING
  This option sets the number of padding characters to 0. This also implies that derived CHARACTER (CHAR) types are not converted to CHARACTER VARYING (VARCHAR) when PADDING CHARACTERS is used.

Note

Oracle recommends that applications always check for SQLDA_CHAR and SQLDA_VARCHAR so that the correctly formatted data is made available to SQL.

  This is the default setting.
- SQL99
  SQL92
  MIA
  SQL89
  SQLV40
  Any of these options will revert to the default semantic for the SQLDA which includes disabling NAMED MARKERS.

Usage Notes

- The ORACLE LEVEL1 and ORACLE LEVEL2 settings are reserved for use by Oracle Corporation. Current behavior of this setting may change with any given release based on requirements of the OCI Services for Rdb component. This setting changes the usage of various SQLDA and SQLDA2 fields.
- Keywords may not be abbreviated and the clauses must be fully specified.
- The SET DIALECT command will implicitly enable NAMED MARKERS if the dialect is changed to either ORACLE LEVEL1 or ORACLE LEVEL2.
- The SET DIALECT command will implicitly disable NAMED MARKERS if the dialect is changed to any dialect other than ORACLE LEVEL1 or ORACLE LEVEL2.
- When NAMED MARKERS are enabled, the contents of the SQLDA and SQLDA2 will reflect one entry for each name. When traditional parameter markers are used, a SQLDA (or SQLDA2) entry will exist for each marker (?) encountered. This change in behavior can simplify the query encoding as well lead to more efficient strategy creation.

Examples

Using the NAMED MARKERS Feature

This example shows that enabling the NAMED MARKERS feature will allow SQL to prompt for one value and the displayed Rdb strategy shows that only one variable is used.

```
-> SET SQLDA 'ENABLE NAMED MARKERS';
-> SELECT LAST_NAME FROM EMPLOYEES WHERE FIRST_NAME = :F_NAME AND LAST_NAME <>
:F_NAME;
in:  [0] typ=449 len=46
out: [0] typ=453 len=14
[SQLDA - reading 1 fields]
-> Alvin
Tables:
  0 = EMPLOYEES
Conjunct: (0.FIRST_NAME = <var0> AND (0.LAST_NAME <> <var0>))
Get     Retrieval sequentially of relation 0:EMPLOYEES
 0/FIRST_NAME/Varchar(42/46): Alvin
[SQLDA - displaying 1 fields]
 0/LAST_NAME: Toliver
[SQLDA - displaying 1 fields]
 0/LAST_NAME: Dement
```

Using the PADDING Feature

The following example shows that the derived type for the named parameter MI is a SQLDA_CHAR (453) of length 1. The input data ('AA') is truncated on assignment and the incorrect results are returned. By adding a small padding, the type is changed to SQLDA_VARCHAR (449) of length 3 and a correct comparison is performed.

```
-> ATTACH 'filename sql$database';
-> SET SQLDA 'enable named markers, nopadding';
-> SELECT LAST_NAME FROM EMPLOYEES WHERE MIDDLE_INITIAL = :MI;
in:  [0] typ=453 len=1
out: [0] typ=449 len=18
[SQLDA - reading 1 fields]
-> AA
[SQLDA - displaying 1 fields]
 0/LAST_NAME: Toliver
[SQLDA - displaying 1 fields]
 0/LAST_NAME: Lengyel
[SQLDA - displaying 1 fields]
```

```
 0/LAST_NAME: Robinson
[SQLDA − displaying 1 fields]
 0/LAST_NAME: Ames
−> SET SQLDA 'padding 2 characters';
−> SELECT LAST_NAME FROM EMPLOYEES WHERE MIDDLE_INITIAL = :MI;
in:  [0] typ=449 len=7
out: [0] typ=449 len=18
[SQLDA − reading 1 fields]
−> AA
−> EXIT;
Enter statement:
```

Note that the VARCHAR requires an extra 4 bytes for the length information in the SQLDA2 used by the Dynamic SQL testing program.

4.1.7 RMU /SHOW VERSION Displays System Architecture and Version

The RMU /SHOW VERSION command has been enhanced to include information about the system architecture and OpenVMS version as shown in the following example:

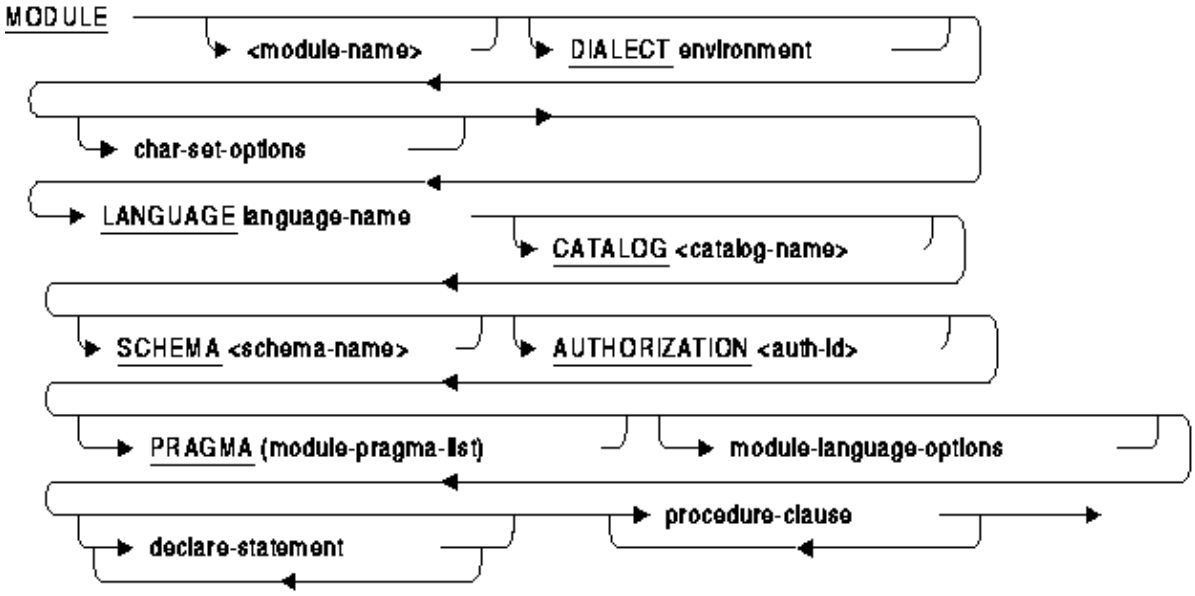
```
$ RMU /SHOW VERSION
Executing RMU for Oracle Rdb V7.2−400 on OpenVMS IA64 V8.3−1H1
$
```

4.1.8 New IDENT Option for SQL Module Language PRAGMA Clause

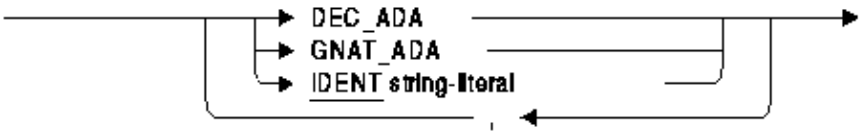
Many OpenVMS compilers allow the programmer to specify the object file IDENT string. This allows tracking of the correct version in linker map files (.map) and within object libraries using the LIBRARIAN command. This release of Oracle Rdb supports setting the IDENT string for the SQL module language.

The IDENT string is specified as part of the PRAGMA clause in the module header.

Syntax



module-pragma-list =



Usage Notes

- By using the PRAGMA clause with IDENT you can record an identification string in the object module generated by the SQL Module language. This IDENT string is recorded by the OpenVMS LINKER in the image itself and can be viewed in the generated MAP file, examined using ANALYZE/OBJECT, and by the LIBRARIAN command when the object module is stored in an object library.
- OpenVMS limits the IDENT string to a 15 octet string. If the string is longer than this (even with trailing spaces) then an error will be reported by the SQL Module Language compiler.
- If the IDENT clause is omitted, then the default version string will default to 'V1.0' as is the practice with many OpenVMS compilers. Prior versions of Oracle Rdb on Integrity systems would only provide the string '0'.

```
Module name:                            "MODSQL$TEST"
Module version:                         "0"
Creation date/time:                     "15-JUN-2009 20:13"
Language name:                          "Oracle Rdb SQL V7.2-351"
```

Examples

The following example shows the use of a PRAGMA clause in a module header to specify the module ident string.

Example 4−1 PRAGMA Clause in the Module Header

```
MODULE         MODSQL$TEST
DIALECT        SQL99
LANGUAGE       C
AUTHORIZATION  SAMPLE_USER
PRAGMA         (IDENT 'V1.2−300')
ALIAS          RDB$DBHANDLE
PARAMETER      COLONS
```

The DCL command ANALYZE/OBJECT can be used to examine the ident string in the object file.

Example 4−2 Examining the IDENT in the Object Module

```
$ sql$mod TEST
$ analyze/object TEST/interactive
This is an OpenVMS IA64 (Elf format) object file

Module Identification Information, in note section 2.

    Module name:                                "MODSQL$TEST"
    Module version:                             "V1.2−300"
    Creation date/time:                         "15−JUN−2009 20:04"
    Language name:                              "Oracle Rdb SQL V7.2−401"
Press RETURN to continue, or enter a period (.) for next file:
<Ctrl/Z>
$
```

Here is similar output from an OpenVMS Alpha system.

```
$ sql$mod TEST
$ analyze/object TEST
   .
   .
   .
This is an OpenVMS Alpha object file

1.  MODULE HEADER (EOBJ$C_EMH), 71 bytes

        structure level: 2
        maximum record size: 4088
        module name: "MODSQL$TEST"
        module version: "V1.0"
        creation   date/time: 16−JUN−2009 11:02
   .
   .
   .
```

This example shows the use of the LIBRARIAN to display the ident strings for object modules in a project object library.

```
$ librarian/list/full project.olb
Directory of ALPHA OBJECT library DISK1:[TESTER]PROJECT.OLB;1 on 16−JUN−2009
11:07:23
```

```
Creation date:  16-JUN-2009 11:07:11      Creator:  Librarian A09-30
Revision date:  16-JUN-2009 11:07:11      Library format:   3.0
Number of modules:      1                 Max. key length:  128
Other entries:          5                 Preallocated index blocks:    213
Recoverable deleted blocks:        0      Total index blocks used:        2
Max. Number history records:      20      Library history records:        0

MODSQL$TEST      Ident V1.2-300         Inserted 16-JUN-2009 11:07:11 5 symbols
```

4.1.9 New Keyword for SQL Module Language /PRAGMA Qualifier

This release of Oracle Rdb adds a new option to the /PRAGMA qualifier. The keyword IDENT can be used to pass a text string to the SQL Module Language compiler to be written to the Object Module Header.

The following example demonstrates the use of the qualifier to establish the generation of the compiler module.

```
$ SQL$MOD TEST/PRAGMA=IDENT="v1.2-32"
```

Usage Notes

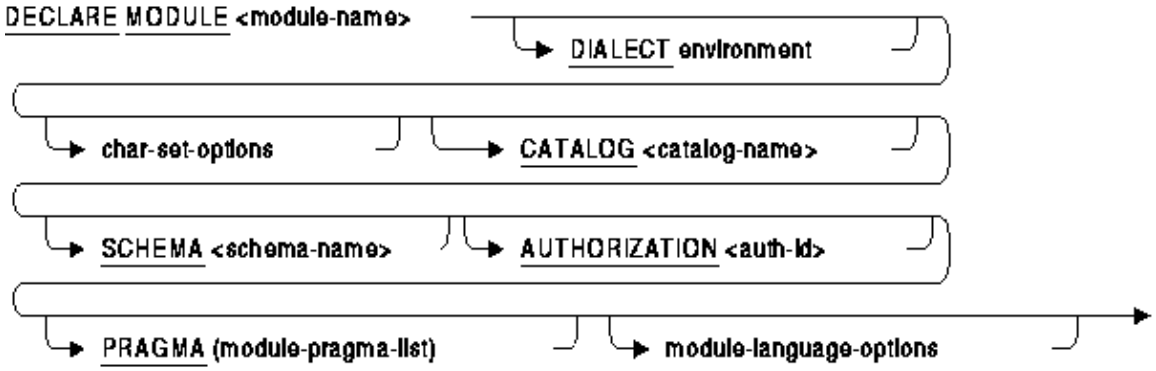
- If the PRAGMA (IDENT ...) clause is used as part of the MODULE header then that value will override any value used on the command line.
- The ANALYZE/OBJECT and LIBRARY commands can be used to display this IDENT string and the value will be displayed in LINKER map files.
- OpenVMS limits the IDENT string to a 15 octet string. If the string is longer than this (even with trailing spaces) then an error will be reported by the SQL Module Language compiler.

4.1.10 New IDENT Option for SQL Precompiler DECLARE MODULE Statement

Many OpenVMS compilers allow the programmer to specify the object file IDENT string. This allows tracking of the correct version in linker map files (.map) and within object libraries using the LIBRARIAN command. This release of Oracle Rdb supports setting the IDENT string for the SQL precompiler module header.

The IDENT string is specified as part of the PRAGMA clause in the module header.

Syntax



module-pragma-list =

Usage Notes

- By using the PRAGMA clause with IDENT, you can record an identification string in the object module generated by the SQL Precompiler. This IDENT string is recorded by the OpenVMS LINKER in the image itself and can be viewed in the generated MAP file, examined using ANALYZE/OBJECT, and by the LIBRARIAN command when the object module is stored in an object library.
- OpenVMS limits the IDENT string to a 15 octet string. If the string is longer than this (even with trailing spaces), then an error will be reported by the SQL Precompiler.
- If the IDENT clause is omitted, then the default version string will default to 'V1.0' as is the practice with many OpenVMS compilers. Prior versions of Oracle Rdb on Integrity systems would only provide the string '0'.

```
Module name:                            "MODSQL$TEST"
Module version:                         "0"
Creation date/time:                     "15-JUN-2009 20:13"
Language name:                          "Oracle Rdb SQL V7.2-351"
```

Examples

The following example shows the use of a PRAGMA clause in a module header to specify the module ident string.

Example 4−3 PRAGMA clause in the DECLARE MODULE statement

```
$ CREATE TEST_HDR.SQL
DECLARE
    MODULE          MODSQL$TEST
    DIALECT         SQL99
    AUTHORIZATION   SAMPLE_USER
    PRAGMA          (IDENT 'V1.2-300')
```

```
;
```

The DCL command ANALYZE/OBJECT can be used to examine the ident string in the object file. Note that the SQL Precompiler generates two object files which are concatenated. Therefore, the ANALYZE will show two MODULE HEADER records, one for the host language (C for example) and one from the SQL Precompiler.

Example 4−4 Examining the IDENT in the object module

```
$ sql$pre/CC TEST TEST_HDR
$ analyze/object TEST/output=SYS$OUTPUT
   .
   .
   .
This is an OpenVMS Alpha object file

175.  MODULE HEADER (EOBJ$C_EMH), 75 bytes
        structure level: 2
        maximum record size: 4088
        module name: "MODSQL$TEST"
        module version: "V1.2−300"
        creation   date/time: 16−JUL−2009 16:50
   .
   .
   .
```

4.1.11 New Keyword for SQL Precompiler PRAGMA Option

This release of Oracle Rdb adds a new keyword to the SQLOPTIONS qualifiers PRAGMA option. The keyword IDENT can be used to pass a text string to the SQL Precompiler to be written to the Object Module Header.

The following example demonstrates the use of the qualifier to establish the generation of the compiler module.

```
$ SQL$PRE/CC TEST/SQLOPTION=(PRAGMA=IDENT="v1.2−32")
```

Usage Notes

- If the PRAGMA (IDENT ...) clause is used as part of the DECLARE MODULE statement, then that value will override any value used on the command line.
- The ANALYZE/OBJECT and LIBRARY commands can be used to display this IDENT string and the value will be displayed in LINKER map files.
- OpenVMS limits the IDENT string to a 15 octet string. If the string is longer than this (even with trailing spaces) then an error will be reported by the SQL precompiler.

4.1.12 RDB_STATS_DATABASE Example Program

Accessing performance information in a tabular fashion for Oracle Rdb databases can often be beneficial. In

particular, stored RMU /SHOW STATISTICS rate information in a database can be utilized to do trend anaysis and historical review of performance indicators.

RDB_STATS_DATABASE is a sample program that reads an RMU /SHOW STATISTICS binary file and converts all statistic values for each sample into a current rate per second. The statistics values are written to a database table named RMU$STATISTICS. If the RMU$STATISTICS table does not exist in the database, it will be created.

To use the RDB_STATS_DATABASE program, create a foreign command symbol with a value of "SQLSAMPLE:RDB_STATS_DATABASExx.EXE" (where xx is the version of Rdb) and pass an output database and an input binary statistics file name. The following example command sequence demonstrates one possible way that statistics can be gathered for one hour and then formatted.

```
$ RDB_STATS_DATABASE := $SQL$SAMPLE:RDB_STATS_DATABASE72
$ RMU /SHOW STATISTICS MFP -
        /NOINTERACTIVE -
        /OUTPUT = 2008-11-16-00-56.STATS -
        /UNTIL = "16-NOV-2008 11:00:00" -
        /TIME = 15
$ RDB_STATS_DATABASE MYDB.RDB 2008-11-16-00-56.STATS
```

This program can be used to capture either "static" data (from a perviously collected binary file) or "real time" data where records are written to the database as they are produced from RMU /SHOW STATISTICS, as in the following example (note that the RDB_STATS_DATABASE example program should be modified when used in this fashion to commit after every record):

```
$ RDB_STATS_DATABASE := $SQL$SAMPLE:RDB_STATS_DATABASE72
$ CREATE /MAILBOX RDB_STATS_DATABASE$MAILBOX -
    /PERMANENT -
    /LOG -
    /BUFFER_SIZE = 65535 -
    /MESSAGE_SIZE = 10000
$ SPAWN RMU /SHOW STATISTICS MFP -
        /NOINTERACTIVE -
        /OUTPUT = RDB_STATS_DATABASE$MAILBOX:
        /UNTIL = "16-NOV-2009 11:00:00" -
        /TIME = 60
$ RDB_STATS_DATABASE MYDB.RDB RDB_STATS_DATABASE$MAILBOX:
```

This example is intended solely to be used as a template for writing your own program. No support for this example template program is expressed or implied.

Oracle Corporation assumes no responsibility for the functionality, correctness or use of this example program. Oracle Corporation reserves the right to change the format and contents of the Oracle Rdb RMU SHOW STATISTICS binary output file at any time without prior notice.

The RDB_STATS_DATABASE example program is comprised of the following source modules found in SQL$SAMPLE:

- RDB_STATS_DATABASExx.C
- RDB_STATS_DATABASE_SQL1_xx.SQLMOD
- RDB_STATS_DATABASE_SQL2_xx.SQLMOD

Compile and link the RDB_STATS_DATABASE example program as follows:

```
$ CC /FLOAT=IEEE RDB_STATS_DATABASExx.C
$ SQL$MOD /FLOAT=IEEE RDB_STATS_DATABASE_SQL1_xx.SQLMOD
$ SQL$MOD /FLOAT=IEEE RDB_STATS_DATABASE_SQL2_xx.SQLMOD
$ LINK RDB_STATS_DATABASExx+−
    RDB_STATS_DATABASE_SQL1_xx+−
    RDB_STATS_DATABASE_SQL2_xx+−
    SQL$USER /LIBRARY
```

4.1.13 RCS Time−Based Cache Sweeping

Previously, the Record Cache Server (RCS) process would perform modified row cache "sweep" operations only when a cache was full (also known as "clogged") with modified rows. Now a database may be configured to perform timed cache sweeps. This feature is intended to help perform "lazy" updates of modified rows to the database from caches without performing a full cache checkpoint operation.

The timer for the periodic cache sweeps is specified with the "SWEEP INTERVAL is numeric−literal seconds" clause of the ALTER DATABASE ... ROW CACHE IS ENABLED statement, as in the following example:

```
ALTER DATABASE FILENAME MF_PERSONNEL
  ROW CACHE IS ENABLED (SWEEP INTERVAL IS 300 SECONDS);
```

The number of slots per cache to sweep is specified with the ALTER CACHE statement. Legal values for "SWEEP INTERVAL" are from 0 seconds (to disable periodic timed sweeps) to 3600 seconds (1 hour).

The RMU /SET ROW_CACHE command accepts a /[NO]SWEEP_INTERVAL=n qualifier as an alternate method to specify the periodic cache sweep timer. /NOSWEEP_INTERVAL disables periodic timed sweeps and /SWEEP_INTERVAL=n can be used to set the timer for the periodic cache sweeps. Legal values for /SWEEP_INTERVAL=n are from 1 second to 3600 seconds (1 hour).

The Record Cache Server (RCS) process log file contains information about periodic row cache "sweep" operations and can be a useful analysis tool.

Default Value

The intended default value for the SWEEP INTERVAL in a database is zero seconds (meaning disabled). It is, however, possible for a database that had originally been created with Oracle Rdb Release 7.0 to have a non−zero value. Customers using the row cache feature are advised to explicitly set the SWEEP INTERVAL parameter to either zero (to disable periodic timed sweeps) or the desired sweep interval period on all databases after upgrading to Release 7.2.4.

Use RMU /SET ROW_CACHE /NOSWEEP_INTERVAL

In Oracle Rdb Release 7.2.4, the "SWEEP INTERVAL is 0 seconds" clause of the ALTER DATABASE ... ROW CACHE IS ENABLED statement may not disable the periodic row cache "sweep" operations. Oracle recommends using the RMU /SET ROW_CACHE /NOSWEEP_INTERVAL command as an alternative. This problem will be corrected in a future release.

Incomplete Display Support

Oracle Rdb Release 7.2.4 does not include complete support for showing a database's periodic row cache "sweep" operation timer value. As a workaround prior to the next Oracle Rdb release, the RMU/DUMP/HEADER/OPTIONS=DEBUG command can be used to display the database parameter RCS_SWEEP_INTERVAL as in the following example:

```
$ RMU/DUMP/HEADER/OPTIONS=DEBUG/OUTPUT=X.X MF_PERSONNEL
$ SEARCH X.X RCS_SWEEP_INTERVAL
RCACHE_CNT = 11.          RCACHE_VBN = 153.         RCS_SWEEP_INTERVAL = 123.
```

4.1.14 RMU Command TSN Keyword and Qualifier Value

RMU commands that accept a TSN keyword or qualifier value now accept input formats as follows:

- A decimal string representing a quadword TSN value
- A hexadecimal string starting with "%X" representing a quadword TSN value
- A two−part decimal string separated by a colon representing a quadword TSN value as high and low longwords

Following are some example uses of input TSN values:

```
$ RMU /DUMP /AFTER_JOURNAL J1.AIJ /FIRST=TSN=54321
$ RMU /DUMP /AFTER_JOURNAL J1.AIJ /FIRST=TSN=123456234253245
$ RMU /DUMP /AFTER_JOURNAL J1.AIJ /FIRST=TSN=%X7655
$ RMU /DUMP /AFTER_JOURNAL J1.AIJ /FIRST=TSN=%X000000715F856AB
$ RMU /DUMP /AFTER_JOURNAL J1.AIJ /FIRST=TSN=0:871251
$ RMU /DUMP /AFTER_JOURNAL J1.AIJ /FIRST=TSN=3:53487
$ RMU /DUMP /AFTER_JOURNAL J1.AIJ /FIRST=TSN=21:653156
```

4.1.15 New Support for RENAME and CREATE SYNONYM Commands

With this release of Oracle Rdb, the RENAME and CREATE SYNONYM commands support INDEX and STORAGE MAP database objects.

- RENAME INDEX changes the name of the index in all system tables.
  A synonym is created using the old index name to reference the new name of the index. This synonym will be used by any query outline that previously referenced the index using the old name. Note that only a single synonym name may exist. Therefore, if you have indices with the same name as another object, then the RENAME INDEX command may fail if creating the synonym detects a duplicate name,
  The command ALTER INDEX ... RENAME TO ... is synonymous with the RENAME INDEX command.
- RENAME STORAGE MAP changes the name of the storage map in all system tables.

If the storage map has a companion function in the RDB$STORAGE_MAPS system module, then that function will also be renamed. A synonym is created using the old function name to reference the new name of the function. This synonym will be used by any other routine, computed by column, automatic column, and so on that referenced the old storage mapping function.
The command ALTER STORAGE MAP ... RENAME TO ... is synonymous with the RENAME STORAGE MAP command.

- CREATE SYNONYM ... FOR INDEX ... is now supported. Synonyms for indices can be created, altered and dropped.
- CREATE SYNONYM ... FOR STORAGE MAP ... is now supported. Synonyms for storage maps can be created, altered and dropped.

The following example shows the result of the RENAME INDEX and RENAME STORAGE MAP commands.

```
SQL> show table (storage maps,index) employees
Information for table EMPLOYEES

Indexes on table EMPLOYEES:
EMPLOYEES_HASH                  with column EMPLOYEE_ID
  No Duplicates allowed
  Type is Hashed Scattered
  Key suffix compression is DISABLED

EMP_EMPLOYEE_ID                 with column EMPLOYEE_ID
  No Duplicates allowed
  Type is Sorted
  Key suffix compression is DISABLED
  Node size  430

EMP_LAST_NAME                   with column LAST_NAME
  Duplicates are allowed
  Type is Sorted
  Key suffix compression is DISABLED

Storage Map for table EMPLOYEES:
     EMPLOYEES_MAP

SQL> rename storage map EMPLOYEES_MAP to EMP_STORAGE_MAP;
SQL> rename index EMPLOYEES_HASH to EMP_ID_HASH;
SQL> show table (storage maps,index) employees
Information for table EMPLOYEES


Indexes on table EMPLOYEES:
EMP_EMPLOYEE_ID                 with column EMPLOYEE_ID
  No Duplicates allowed
  Type is Sorted
  Key suffix compression is DISABLED
  Node size  430

EMP_ID_HASH                     with column EMPLOYEE_ID
  No Duplicates allowed
  Type is Hashed Scattered
  Key suffix compression is DISABLED

EMP_LAST_NAME                   with column LAST_NAME
  Duplicates are allowed
  Type is Sorted
  Key suffix compression is DISABLED
```

```
Storage Map for table EMPLOYEES:
     EMP_STORAGE_MAP

SQL> show storage map
User Storage Maps in database with filename mf_personnel_sql
     CANDIDATES_MAP
     COLLEGES_MAP
     DEGREES_MAP
     DEPARTMENTS_MAP
     EMP_STORAGE_MAP
     JOBS_MAP
     JOB_HISTORY_MAP
     SALARY_HISTORY_MAP
     WORK_STATUS_MAP
SQL> show index
User indexes in database with filename mf_personnel_sql
     COLL_COLLEGE_CODE
     DEG_COLLEGE_CODE
     DEG_EMP_ID
     DEPARTMENTS_INDEX
     EMP_EMPLOYEE_ID
     EMP_ID_HASH
     EMP_LAST_NAME
     JH_EMPLOYEE_ID
     JOB_HISTORY_HASH
     SH_EMPLOYEE_ID
     EMPLOYEES_HASH                  A synonym for index EMP_ID_HASH
SQL> show system function
Functions in database with filename mf_personnel_sql
     CANDIDATES_MAP
     COLLEGES_MAP
     DEGREES_MAP
     DEPARTMENTS_MAP
     EMP_STORAGE_MAP
     JOBS_MAP
     JOB_HISTORY_MAP
     SALARY_HISTORY_MAP
     WORK_STATUS_MAP
     EMPLOYEES_MAP                   A synonym for function EMP_STORAGE_MAP
SQL>
```

Chapter 5
Enhancements And Changes Provided in Oracle Rdb Release 7.2.3.5

5.1 Enhancements And Changes Provided in Oracle Rdb Release 7.2.3.5

5.1.1 AIJ Extend Additional Information In Operator Notification New Feature

Bug 8286207

In order to help understand the impact of AIJ extension operations on the system, a new optional feature can be used to display additional OPCOM messages and perform validation (read checking of initialized data) during AIJ extend operations.

If the logical name RDM$BIND_AIJ_EXTEND_ADDITIONAL_INFO is defined to a value of "1" (Oracle recommends that this logical be defined system−wide if you intend to take advantage of this feature), and if the database is configured to send operator notifications, additional OPCOM messages will be generated during an AIJ extend operation. This optional feature also includes a validation of the initialization pattern written to the AIJ file. If an invalid pattern is detected, the process performing the extension will bugcheck.

The following example shows the format of the additional messages that indicate the old and new physical EOF locations, the number of blocks of the AIJ file being initialized, the number of IO operations required for the initialization and the ID of the process performing the extension and initialization.

```
%%%%%%%%%%  OPCOM  25−MAR−2009 02:31:45.95  %%%%%%%%%%
Message from user SUPERDOME on BRDBRY
Oracle Rdb V7.2−340 Event Notification for Database
$1$DGA301:[SUPERDOME.V72]FOO.RDB;3

AIJ journal 512 block extension in progress (new size is 95552 blocks)

%%%%%%%%%%  OPCOM  25−MAR−2009 02:31:45.95  %%%%%%%%%%
Message from user SUPERDOME on BRDBRY
Oracle Rdb V7.2−340 Event Notification for Database
$1$DGA301:[SUPERDOME.V72]FOO.RDB;3

AIJ new PEOF = 95616, old PEOF = 95040, init count = 576, PID = 0000049E

%%%%%%%%%%  OPCOM  25−MAR−2009 02:31:45.95  %%%%%%%%%%
Message from user SUPERDOME on BRDBRY
Oracle Rdb V7.2−340 Event Notification for Database
$1$DGA301:[SUPERDOME.V72]FOO.RDB;3

AIJ initialize IO count = 3, validating VBN 95041 to 95616, PID = 0000049E
```

5.1.2 Default Behavior Change, New Syntax for RMU/RECOVER/CONFIRM

Bugs 7656967 and 8242546

For RMU/RECOVER/NOCONFIRM, where the user cannot be prompted for a decision on how to proceed, the default behavior in the case where the next Oracle Rdb AIJ file to be rolled forward is missing or is not specified in the correct sequence, has been changed. This also affects RMU/RECOVER executed in a batch job where /NOCONFIRM is the default and /CONFIRM cannot be specified unless it is used with the new parameters /CONFIRM=ABORT or /CONFIRM=CONTINUE.

The old behavior for RMU/RECOVER/NOCONFIRM and RMU/RECOVER executed in batch jobs was to assume the user wanted to continue rolling forward if there was an AIJ file sequence gap. This could cause loss of data and invalid indexes due to not rolling forward the database transactions contained in the missing AIJ file. If the user did not do a full RMU/VERIFY of the database following the recovery, he would not be immediately aware of these problems.

To protect the integrity of Rdb databases, the new default behavior for RMU/RECOVER/NOCONFIRM and RMU/RECOVER executed in batch jobs is to terminate the RMU/RECOVER at the point where the out−of−sequence AIJ file is detected. To override this new default behavior, the user can continue to roll forward and ignore the missing AIJ file either by specifying the existing command syntax RMU/RECOVER/CONFIRM to get a prompt on whether to continue rolling forward if there is an AIJ sequence gap, or by specifying the new syntax RMU/CONFIRM=CONTINUE if he does not want the prompt or is executing the RMU/RECOVER in a batch job.

The user can specify the new command syntax RMU/CONFIRM=ABORT if he wants to terminate the RMU/RECOVER at the point where the out−of−sequence AIJ file is detected. This is the new default behavior if /NOCONFIRM is specified or if RMU/RECOVER is executed in a batch job. For interactive recoveries, /CONFIRM, which prompts the user, will continue to be the default.

The new syntax added to the RMU/RECOVER command is the following.

- Do not prompt the user if a sequence gap is detected on the next AIJ file to be rolled forward but ignore the missing AIJ file and continue rolling forward.

  ```
  /CONFIRM=CONTINUE
  ```
- Do not prompt the user if a sequence gap is detected on the next AIJ roll forward but end the database recover at this point. This is the same as the new default behavior for RMU/RECOVER/NOCONFIRM and RMU/RECOVER in batch.

  ```
  /CONFIRM=ABORT
  ```

The default for interactive recoveries continues to be /CONFIRM which prompts the user to see if he wants to continue.

The following example shows the new "/CONFIRM=CONTINUE" syntax used so that RMU/RECOVER will continue rolling forward if a sequence gap is detected. This used to be the default behavior if /NOCONFIRM was specified or for batch jobs but is no longer the default behavior.

```
RMU/RECOVER/CONFIRM=CONTINUE/LOG/ROOT=user$test:foo faijbck1,faijbck2,faijbck4
%RMU−I−LOGRECDB, recovering database file DEVICE:[DIRECTORY]FOO.RDB;1

%RMU−I−LOGOPNAIJ, opened journal file DEVICE:[DIRECTORY]FAIJBCK4.AIJ;1
 at 25−FEB−2009 17:26:04.00
%RMU−W−AIJSEQAFT, incorrect AIJ file sequence 8 when 7 was expected
%RMU−I−AIJONEDONE, AIJ file sequence 8 roll−forward operations completed
%RMU−I−LOGRECOVR, 1 transaction committed
%RMU−I−LOGRECOVR, 0 transactions rolled back
```

```
%RMU-I-LOGRECOVR, 0 transactions ignored
%RMU-I-AIJNOACTIVE, there are no active transactions
%RMU-I-AIJSUCCES, database recovery completed successfully
%RMU-I-AIJNXTSEQ, to continue this AIJ file recovery, the sequence number
 needed will be 9
%RMU-I-AIJALLDONE, after-image journal roll-forward operations completed
%RMU-I-LOGSUMMARY, total 3 transactions committed
%RMU-I-LOGSUMMARY, total 0 transactions rolled back
%RMU-I-LOGSUMMARY, total 0 transactions ignored
%RMU-I-AIJSUCCES, database recovery completed successfully
%RMU-I-AIJFNLSEQ, to start another AIJ file recovery, the sequence number
 needed will be 9
%RMU-I-AIJNOENABLED, after-image journaling has not yet been enabled
```

The following example shows the new "/CONFIRM=ABORT" syntax used so that RMU/RECOVER will not continue rolling forward if a sequence gap is detected. This is now the default behavior if /NOCONFIRM is specified or for batch jobs. Note that the exit status of RMU will be "%RMU-E-AIJRECESQ" if the recovery is aborted due to a sequence gap. It is always a good policy to check the exit status of RMU, especially when executing RMU in batch jobs.

```
RMU/RECOVER/CONFIRM=ABORT/LOG/ROOT=user$test:foo faijbck1,faijbck2,faijbck4
%RMU-I-LOGRECDB, recovering database file DEVICE:[DIRECTORY]FOO.RDB;1

%RMU-I-LOGOPNAIJ, opened journal file DEVICE:[DIRECTORY]FAIJBCK4.AIJ;1
 at 25-FEB-2009 17:27:42.29
%RMU-W-AIJSEQAFT, incorrect AIJ file sequence 8 when 7 was expected
%RMU-E-AIJRECESQ, AIJ roll-forward operations terminated due to sequence error
%RMU-I-AIJALLDONE, after-image journal roll-forward operations completed
%RMU-I-LOGSUMMARY, total 2 transactions committed
%RMU-I-LOGSUMMARY, total 0 transactions rolled back
%RMU-I-LOGSUMMARY, total 0 transactions ignored
%RMU-I-AIJFNLSEQ, to start another AIJ file recovery, the sequence number
 needed will be 7
%RMU-I-AIJNOENABLED, after-image journaling has not yet been enabled
```

This change has been made in Oracle Rdb Release 7.2.3.5.

Chapter 6
Enhancements And Changes Provided in Oracle Rdb Release 7.2.3.2

6.1 Enhancements And Changes Provided in Oracle Rdb Release 7.2.3.2

6.1.1 New SQL Functions Added

Bug 7365167

This release of Oracle Rdb adds new functions to the SYS$LIBRARY:SQL_FUNCTIONS72.SQL script. To replace the existing library of functions first use the SQL_FUNCTIONS_DROP72.SQL script and then reapply using SQL_FUNCTIONS72.SQL.

Description

- ACOS returns the arc cosine of n. The argument n must be in the range of −1 to 1, and the function returns a DOUBLE PRECISION value in the range of 0 to pi, expressed in radians.
  If the passed expression results in NULL, then the result of ACOS will be NULL.
  The following example returns the arc cosine of .3:

```
SQL> SELECT ACOS(.3) "Arc_Cosine" FROM Rdb$DATABASE;
              Arc_Cosine
  1.266103672779499E+000
1 row selected
```

- ACOSH returns the hyperbolic arc cosine of n. The argument n must be equal to or greater than 1. The function returns a DOUBLE PRECISION value.
  If either passed expression results in NULL, then the result of ACOSH will be NULL.

```
SQL> SELECT ACOSH(1.0) "Hyperbolic Arc Cosine" FROM Rdb$DATABASE;
   Hyperbolic Arc Cosine
  0.000000000000000E+000
1 row selected
```

- ASIN returns the arc sine of n. The argument n must be in the range of −1 to 1, and the function returns a DOUBLE PRECISION value in the range of −pi/2 to pi/2, expressed in radians.
  If the passed expression results in NULL, then the result of ASIN will be NULL.
  The following example returns the arc sine of .3:

```
SQL> SELECT ASIN(.3) "Arc_Sine" FROM Rdb$DATABASE;
                Arc_Sine
  3.046926540153975E-001
1 row selected
```

- ASINH returns the hyperbolic arc sine of n. The function returns a DOUBLE PRECISION value.
  If either passed expression results in NULL, then the result of ASINH will be NULL.

```
SQL> SELECT ASINH (-90.0) "Hyperbolic Arc Sine" FROM Rdb$DATABASE;
     Hyperbolic Arc Sine
 -5.192987713658941E+000
1 row selected
```

- ATAN returns the arc tangent of n. The argument n can be in an unbounded range and returns a value in the range of −pi/2 to pi/2, expressed in radians.
  If the passed expression results in NULL, then the result of ATAN will be NULL.
  The following example returns the arc tangent of .3:

```
SQL> SELECT ATAN(.3) "Arc_Tangent" FROM Rdb$DATABASE;
             Arc_Tangent
  2.914567944778671E-001
1 row selected
```

- ATAN2 returns the arc tangent of n and m. The argument n can be in an unbounded range and returns a value in the range of −pi to pi, depending on the signs of n and m, expressed in radians.
  ATAN2(n,m) is the same as ATAN(n/m).
  If either passed expression results in NULL, then the result of ATAN2 will be NULL.

```
SQL> SELECT ATAN2(.3, .2) "Arc_Tangent2" FROM Rdb$DATABASE;
            Arc_Tangent2
  9.827937232473291E-001
1 row selected
```

- ATANH returns the hyperbolic arc tangent of n (in radians). The argument n must be in the range of −1 to 1, and the function returns a DOUBLE PRECISION value.
  If either passed expression results in NULL, then the result of ATANH will be NULL.

```
SQL> SELECT ATANH(0.905148254) "Hyperbolic Arc Tangent" FROM Rdb$DATABASE;
  Hyperbolic Arc Tangent
  1.500000001965249E+000
1 row selected
```

- BITAND computes an AND operation on the bits of expr1 and expr2, both of which must resolve to integers, and returns an integer. This function is commonly used with the DECODE function, as illustrated in the example that follows.
  If the passed expression results in NULL, then the result of BITAND will be NULL.
  The first bit of the mask stored in the column RDB$FLAGS of the table Rdb$RELATIONS indicates that this relation is a view definition. This query displays the names of any views in the database.

Example 6−1 Checking bits in RDB$FLAGS column

```
SQL> -- Which objects in Rdb$RELATIONS are views?
SQL> select rdb$relation_name from rdb$relations where bitand(rdb$flags, 1) = 1;
 RDB$RELATION_NAME
 RDBVMS$COLLATIONS
 RDBVMS$INTERRELATIONS
 RDBVMS$PRIVILEGES
 RDBVMS$RELATION_CONSTRAINTS
 RDBVMS$RELATION_CONSTRAINT_FLDS
 RDBVMS$STORAGE_MAPS
 RDBVMS$STORAGE_MAP_AREAS
 RDBVMS$TRIGGERS
8 rows selected
```

This example uses the result of the BITAND in a DECODE list to display attributes of an Rdb database.

Example 6−2 Using BITAND with DECODE

```
SQL> select
cont>     DECODE (BITAND (rdb$flags,1), 0, 'No Dictionary', 'Dictionary'),
cont>     DECODE (BITAND (rdb$flags,2), 0, 'ACL Style', 'ANSI Style'),
cont>     DECODE (BITAND (rdb$flags,64), 0, 'No Multischema', 'Multischema')
cont> from
cont>     Rdb$DATABASE;
```

```
 No Dictionary   ACL Style    Multischema
1 row selected
SQL>
```

- COT returns the cotangent of n. The function returns a DOUBLE PRECISION value. If either passed expression results in NULL, then the result of COT will be NULL.

```
SQL> SELECT COT (3.14159265358979/4) "Cotangent" FROM Rdb$DATABASE;
               Cotangent
  1.000000000000002E+000
1 row selected
```

6.1.2 RMU /SHOW STATISTICS Enhanced LogMiner Information Display

The RMU /SHOW STATISTICS "LogMiner Information" display has been enhanced to include additional information about the state of Continuous LogMiner processes.

Information displayed includes the following fields (note that the terminal screen width must be greater than 80 columns to display all fields):

- Process.ID − The Process and Stream ID of the Continuous LogMiner process
- State − The current Continuous LogMiner process state:
  - ♦ Inactive
  - ♦ Hibernating
  - ♦ Polling
  - ♦ Extracting
- SeqNum − The AIJ sequence number currently being read
- CurrVBN − The disk block number currently being processed
- Actv − The number of active transactions being processed
- MQP_VNO − The AIJ sequence number location of the last micro−quiet point detected
- MQP_VBN − The disk block number location of the last micro−quiet point detected
- MQP_TSN − The transaction sequence number of the last micro−quiet point detected
- LastTSN − The transaction sequence number of the last transaction

6.1.3 RMU /SHOW STATISTICS "Checkpoint Statistics" New Counters

Previously, on the "Checkpoint Statistics" screen of the "RMU /SHOW STATISTICS" utility, two of the possible checkpoint conditions were not being captured for display. These conditions now displayed are:

- "Clear" indicates that a process's checkpoint information is to be cleared in the root file.
- "Initial" indicates that a process has performed its initial checkpoint to establish a checkpoint starting location.

Aggregate Total

Because some of the checkpoint conditions can occur in combination, the sum total of all of the possible checkpoint types may exceed the aggregate total "checkpoints" value displayed.

6.1.4 SQL Enhancements: Allowing Optional Correlation Names

Bug 6847694

This release of Oracle SQL improves compatibility with Oracle RDBMS SQL. In particular, some SQL syntax sent via OCI Services for Rdb was rejected in prior releases.

Rdb SQL contains the following enhancements:

1. Oracle RDBMS does not require the correlation name for a derived table (which can be deeply nested). However, in prior releases, Oracle RDB would reject this syntax.

   ```
   SQL> select *
   cont> from (select last_name, first_name from candidates);
   %SQL-F-CORNAMREQ, A correlation name is required for a derived table
   SQL>
   SQL> select *
   cont> from (select last_name, first_name
   cont>       from (select last_name, first_name from candidates));
   %SQL-F-CORNAMREQ, A correlation name is required for a derived table
   ```

   Oracle Rdb now also supports an optional correlation name.

   ```
   SQL> select *
   cont> from (select last_name, first_name from candidates);
    LAST_NAME        FIRST_NAME
    Wilson           Oscar
    Schwartz         Trixie
    Boswick          Fred
   3 rows selected
   SQL>
   SQL> select *
   cont> from (select last_name, first_name
   cont>       from (select last_name, first_name from candidates));
    LAST_NAME        FIRST_NAME
    Wilson           Oscar
    Schwartz         Trixie
    Boswick          Fred
   3 rows selected
   ```
2. Oracle RDBMS allows an optional correlation name for INSERT targets. Oracle Rdb did not permit a correlation name in this location.

   ```
   SQL> create table fred (a integer);
   SQL> insert into fred f (a) values (1);
   insert into fred f (a) values (1);
                          ^
   %SQL-W-LOOK_FOR_STT, Syntax error, looking for:
   %SQL-W-LOOK_FOR_CON,            AS, RETURNING, PLACEMENT, ;,
   ```

```
%SQL-F-LOOK_FOR_FIN,    found VALUES instead
```

Oracle Rdb now also supports an optional correlation name.

```
SQL> create table fred (a integer);
SQL> insert into fred f (a) values (1);
1 row inserted
```

The correlation name is useful for the RETURNING clause.

```
SQL> insert into fred f (a) values (2) returning f.a;
           A
           2
1 row inserted
```

These problems have been corrected in Oracle Rdb Release 7.2.3.2.

6.1.5 Performance Enhancements With Internal Lock Data Structures

In order to help performance for certain classes of applications that lock a large number of records within a transaction, several optimizations have been implemented:

- An internal hash table used to access a list of record locks owned by a database user has been increased in size to help speed access to entries within the table.
- An internal data structure used in conjunction with the hash table is now allocated in larger segments to reduce the number of memory allocations.

6.1.6 Change in Frequency of Cardinality Updates Might be Observed

With this release, Oracle Rdb changed slightly the frequency at which table and index cardinality changes, or workload column group rows (RDB$WORKLOAD) were flushed to the system tables. Some users of RMU/UNLOAD/AFTER_IMAGE or RMU/OPTIMIZE/AFTER_IMAGE may observe one more or one less transaction being processed than in prior releases. The reason for this difference is that Rdb attempts to update the system tables on the tail end of a user's READ WRITE transaction and the change in frequency may or may not require Rdb to start an additional transaction during the DISCONNECT processing in which to write back the final statistics.

6.1.7 New Interactive SQL Statements

This release of Oracle Rdb adds new Interactive SQL statements to improve compatibility with SQL*plus.

- SET FEEDBACK { ON | OFF | n }
  The existing SET FEEDBACK statement now supports a numeric option that defines the threshold at which reported line counters are displayed.

```
SQL> set feedback 2
SQL> select * from work_status;
 STATUS_CODE   STATUS_NAME   STATUS_TYPE
 0             INACTIVE      RECORD EXPIRED
 1             ACTIVE        FULL TIME
 2             ACTIVE        PART TIME
3 rows selected
SQL> set feedback 4
SQL> select * from work_status;
 STATUS_CODE   STATUS_NAME   STATUS_TYPE
 0             INACTIVE      RECORD EXPIRED
 1             ACTIVE        FULL TIME
 2             ACTIVE        PART TIME
```

- SET LINESIZE n
  The SET LINESIZE command is synonymous with the SET LINE LENGTH command.
- SET PAGESIZE n
  The SET PAGESIZE command is synonymous with the SET PAGE LENGTH command.
- SET TIMING { ON | OFF }
  The SET TIMING enables a single line report of used CPU and Elapsed time for each successful SQL statement or command. This is shown in the following example.

```
SQL> start transaction;
SQL> set timing on;
SQL> select count(*)
cont> from employees
cont>      inner join job_history using (employee_id)
cont>      inner join salary_history using (employee_id)
cont>      inner join departments using (department_code)
cont>      inner join jobs using (job_code)
cont>      left outer join resumes using (employee_id)
cont>      left outer join degrees using (employee_id)
cont>      left outer join colleges using (college_code)
cont>
cont> ;

        3871
1 row selected
Timing: Elapsed:    0 00:00:00.82 Cpu:    0 00:00:00.16
SQL> set timing off;
SQL> commit;
```

Chapter 7
Enhancements And Changes Provided in Oracle Rdb Release 7.2.3.0

7.1 Enhancements And Changes Provided in Oracle Rdb Release 7.2.3.0

7.1.1 Optional Run−Time Routine Native Compiler on I64 Enabled By Default

On Alpha and VAX systems, Oracle Rdb generates, at run time, callable subroutines of native machine instructions. These subroutines are executed during the processing of database insert/update/retrieval operations.

Oracle Rdb on I64 systems uses a hardware−portable instruction interpreter. This allowed rapid development and deployment of Oracle Rdb on Integrity OpenVMS systems. Release 7.2.1.1 of Oracle Rdb introduced a second pass optimization that converts these portable instructions to native I64 machine code for enhanced performance. This feature is enabled by default starting with Release 7.2.3 of Oracle Rdb.

To disable the run−time compiler, define RDMS$BIND_CODE_OPTIMIZATION to a value of "0" or use the flag "CODE_OPTIMIZATION(0)". The run−time compiler is enabled by default. To re−enable, if needed, the run−time compiler on I64 systems, either deassign the logical name RDMS$BIND_CODE_OPTIMIZATION or define the logical name RDMS$BIND_CODE_OPTIMIZATION to a value of "2" or use the flag "CODE_OPTIMIZATION(2)". This logical name and flag have no effect on Alpha systems. The following examples show use of the logical name and the flag to enable and disable and show the status of the optional run−time compiler on I64 systems.

```
! To disable:
$ DEFINE RDMS$BIND_CODE_OPTIMIZATION 0
$ DEFINE RDMS$SET_FLAGS "CODE_OPTIMIZATION(0)"
SQL> SET FLAGS 'CODE_OPTIMIZATION(0)';

! To enable:
$ DEFINE RDMS$BIND_CODE_OPTIMIZATION 2
$ DEFINE RDMS$SET_FLAGS "CODE_OPTIMIZATION(2)"
SQL> SET FLAGS 'CODE_OPTIMIZATION(2)';

! Show current setting enabled and disabled:
SQL> ATT 'FI MF_PERSONNEL';
SQL> SET FLAGS 'CODE_OPTIMIZATION(2)';
SQL> SHOW FLAGS

Alias RDB$DBHANDLE:
Flags currently set for Oracle Rdb:
   PREFIX,WARN_DDL,INDEX_COLUMN_GROUP,MAX_SOLUTION
   ,MAX_RECURSION(100),REFINE_ESTIMATES(127)
   ,CODE_OPTIMIZATION(2),NOBITMAPPED_SCAN

SQL> SET FLAGS 'CODE_OPTIMIZATION(0)';
SQL> SHOW FLAGS

Alias RDB$DBHANDLE:
Flags currently set for Oracle Rdb:
   PREFIX,WARN_DDL,INDEX_COLUMN_GROUP,MAX_SOLUTION
   ,MAX_RECURSION(100),REFINE_ESTIMATES(127),NOBITMAPPED_SCAN
SQL>
```

There is CPU overhead required to compile a subroutine and not every subroutine can be compiled, at present, into native Itanium routines. Additional virtual memory is required for the functionality of the optional run−time compiler.

If you detect a difference in the functionality or behavior of Oracle Rdb when this feature is enabled, you may define the logical name to revert to the prior (interpreted) execution model.

7.1.2 Temporary Table Improvements

In prior versions of Oracle Rdb, all temporary table data structures were always allocated in 32−bit process (P0) virtual address space. In several cases, the 1GB size of P0 space drastically limited the number of rows that could be stored or would result in unexpected errors when deleting or modifying rows. Additionally, performance when accessing rows within temporary tables would degrade as additional rows were inserted.

The following enhancements have been made to address these cases:

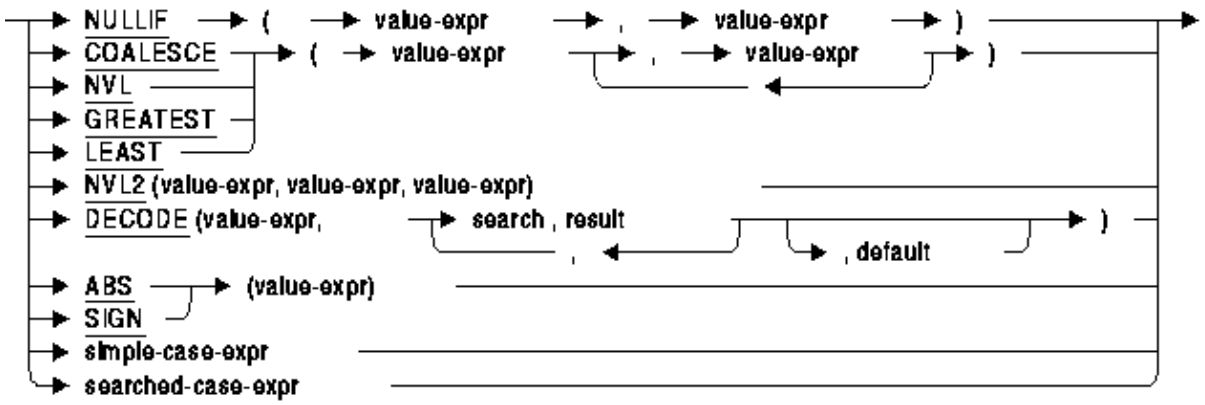
- With the exception of the actual data rows themselves, most of the data structures related to temporary tables have been moved to 64−bit process (P2) virtual address space. The data content remains in 32−bit address space.
- Use of the logical name RDMS$TTB_HASH_SIZE to control the size of an internal hash table may no longer be required for consistent performance. The hash table is now initially sized larger than it was in the past and will automatically extend as needed to accommodate the rows in the temporary table with more consistent performance.

7.1.3 New SIGN Builtin Function

This release of Oracle Rdb enhances the Conditional expressions with support for the SIGN builtin function.

Syntax

conditional-expr =



Usage Notes

- SIGN returns an INTEGER value.
- SIGN accepts any numeric (fixed or floating) or interval value expression.
- If the value expression evaluates to NULL, then the SIGN function returns NULL.
- If the value expression evaluates to a negative value, then SIGN returns −1; if the value is positive then SIGN returns 1; otherwise a zero will be returned.
- The SQL_FUNCTIONS script continues to add the SIGN function to the database. However, this function is now deprecated and is retained only for backward compatibility with applications built using that function.

Examples

This example computes delayed departures from the LAYOVER table.

Example 7−1 Using SIGN Builtin Function

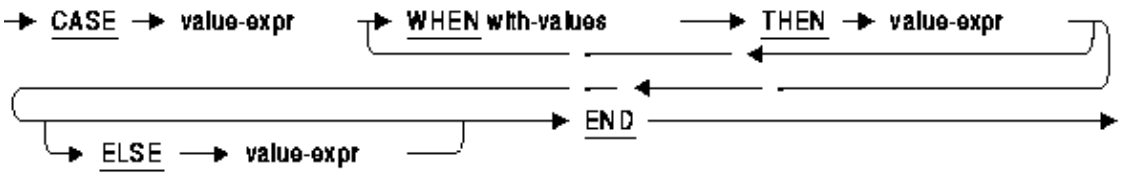
```
SQL> select arr_date,
cont>       dep_date,
cont>       DECODE (SIGN ((dep_date − arr_date) day(9)),
cont>           −1, 'date error − can not depart before arrival',
cont>            0, 'same day departure',
cont>            1, 'delayed')
cont> from LAYOVER;

 2005−12−22   2006−01−20   delayed
 2005−12−23   2005−12−25   delayed
 2006−01−30   2006−02−01   delayed
 2006−02−06   2006−02−09   delayed
 2006−01−24   2006−01−26   delayed
 2006−02−02   2006−02−19   delayed
 2007−02−10   2007−02−16   delayed
 2007−02−20   2007−02−26   delayed
 2007−05−29   2007−06−08   delayed
 2007−06−12   2007−06−26   delayed
 2007−05−15   2007−05−21   delayed
 2007−09−10   2007−09−14   delayed
 2007−09−04   2007−09−06   delayed
 2007−09−19   2007−09−20   delayed
 2007−09−21   2007−09−24   delayed
15 rows selected
```

7.1.4 Enhanced Simple CASE Expression

This release of Oracle Rdb enhances the Simple CASE expression to support value lists and partial predicates.

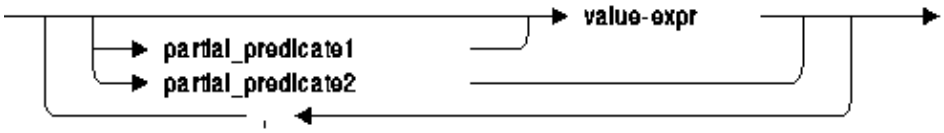
Syntax

simple-case-expr =



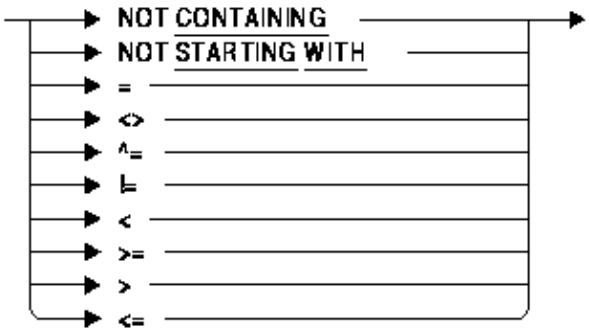
---

with-values =



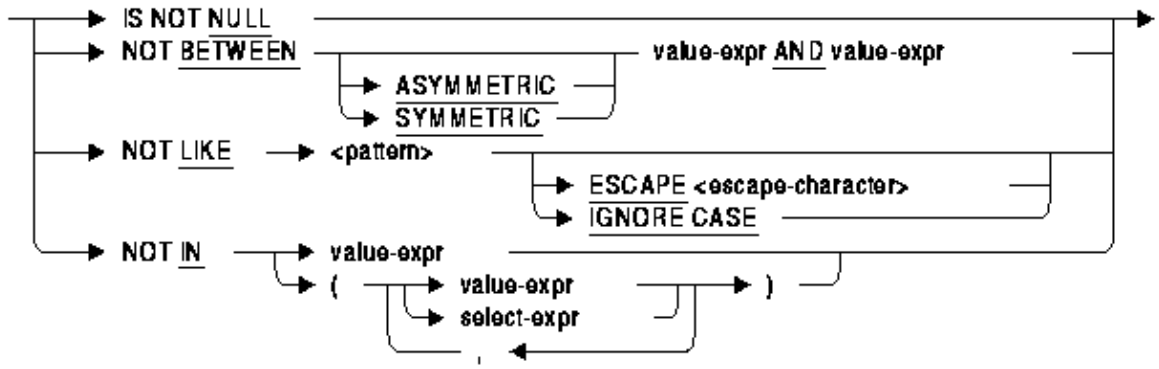
---

partial_predicate1 =



---

partial_predicate2 =



Usage Notes

- The WHEN clause can now include a list of value expressions. In prior versions, the WHEN clause was limited to just one value expression. Each value expression is separated by a comma.
- The WHEN clause can now include partial predicates. In prior versions, the default operator that was applied to the case primary expression of each WHEN value expression was equality (=). With this release, this can be overridden. Multiple partial predicates can also be listed.
  A WHEN clause partial predicate can begin with one of the following operators: IS NULL, IS NOT NULL, BETWEEN, NOT BETWEEN, IN, NOT IN, LIKE, NOT LIKE, STARTING WITH, NOT STARTING WITH, CONTAINING, NOT CONTAINING, <=, <, <>, >=, >, ^=, !=, and =.
- If the WHEN clause includes the value expression NULL, it assumes that this is equivalent to the IS NULL operator.

Examples

This example calculates the calendar quarter using a CASE expression.

Example 7−2 Using the IN clause as a partial predicate

```
SQL> select
cont>     salary_end,
cont>     case extract (month from salary_end)
cont>         when in (1, 2, 3) then 'Q1'
cont>         when in (4, 5, 6) then 'Q2'
cont>         when in (7, 8, 9) then 'Q3'
cont>         when in (10, 11, 12) then 'Q4'
cont>         else 'UNKNOWN'
cont>     end quarter
cont> from salary_history
cont> where employee_id = '00164'
cont> order by salary_start;
 SALARY_END    QUARTER
  2-Mar-1981   Q1
 21-Sep-1981   Q3
 14-Jan-1983   Q1
 NULL          UNKNOWN
4 rows selected
```

```
SQL>
```

A simple list of values could also be used to achieve the same result.

```
SQL> select
cont>     salary_end,
cont>     case extract (month from salary_end)
cont>         when 1, 2, 3 then 'Q1'
cont>         when 4, 5, 6 then 'Q2'
cont>         when 7, 8, 9 then 'Q3'
cont>         when 10, 11, 12 then 'Q4'
cont>         else 'UNKNOWN'
cont>     end quarter
cont> from salary_history
cont> where employee_id = '00164'
cont> order by salary_start;
 SALARY_END    QUARTER
  2-Mar-1981   Q1
 21-Sep-1981   Q3
 14-Jan-1983   Q1
 NULL          UNKNOWN
4 rows selected
SQL>
```

This example shows the use of partial predicates to process the LAST_NAME column and provide a customized ordering of the EMPLOYEES table.

Example 7-3 Using Partial predicates in the CASE expression

```
SQL> select
cont>     concat (trim (first_name), ' ',
cont>             nvl2 (middle_initial,
cont>                   concat (middle_initial,'.',last_name),
cont>                   last_name)) as names
cont> from employees
cont> order by
cont>     case last_name
cont>         when starting with 'Mc'
cont>         then
cont>             substring (last_name from 3)
cont>         when starting with 'Mac'
cont>         then
cont>             substring (last_name from 4)
cont>         when containing ''''
cont>         then
cont>             substring (last_name from position (''''' in last_name)+1)
cont>         else
cont>             last_name
cont>     end;
 NAMES
 Louie A.Ames
 Aruwa D'Amico
 Leslie Q.Andriola
 Joseph Y.Babbin
 Dean G.Bartlett
   .
   .
   .
```

```
 Brian Wood
 Al F.Ziemke
100 rows selected
```

This example shows the use of value lists to make the simple case more readable. Using input values for the month and year, it returns the number of days in that month.

Example 7−4 Using lists of values in the Simple CASE expression

```
SQL> set flags 'trace';
SQL> begin
cont> declare :year_val, :month_val int;
cont> set :year_val = extract (year from current_date);
cont> for :month_val in 1 to 12
cont> do
cont>     trace
cont>         case :month_val
cont>             when 1, 3, 5, 7, 8, 10, 12 then
cont>                 31
cont>             when 4, 6, 9, 11 then
cont>                 30
cont>             when 2 then
cont>                 DECODE (MOD (:year_val, 4), 0, 29, 28)
cont>             else
cont>                 NULL
cont>         end;
cont> end for;
cont> end;
~Xt: 31
~Xt: 29
~Xt: 31
~Xt: 30
~Xt: 31
~Xt: 30
~Xt: 31
~Xt: 31
~Xt: 30
~Xt: 31
~Xt: 30
~Xt: 31
```

7.1.5 Changes in Generated Query Outline ID

This release of Oracle SQL has added various optimizations to the Simple CASE expression as well as the DECODE and ABS functions. These changes might, in some cases, change the generated intermediate query. This will have no effect on the query results but will change the query id generated for a query outline. In such cases, the query outline should be re−created to ensure matching by the new outline id.

Alternately, queries using these query outlines can be modified to use the OPTIMIZE USING clause to use the name of the query outline. Then such changes as described here do not affect the selection of the query outline. If a query outline is used in conjunction with the RMU Unload command then the /OPTIMIZE=USING clause can be used to specify the name of the query outline for use by RMU.

Note

This query outline id is also displayed by the SET FLAGS 'WATCH_OPEN' flag as shown in this example:

```
SQL> set flags 'watch_open';
SQL> select decode (employee_id, '00164', 'Toliver', 'Others') from employees;
~Xo: Start Request ECB31E522CC71247B16B18660AD42F1D (unnamed)
   .
   .
   .
```

This is the query id generated by Oracle Rdb V7.1.

```
SQL> select decode (employee_id, '00164', 'Toliver', 'Others') from employees;
~Xo: Start Request 77D353270F7842D60FCA3D6CC11378D3 (unnamed)
   .
   .
   .
```

7.1.6 ALTER INDEX ... MAINTENANCE IS ENABLED DEFERRED Syntax is Now Active

Bug 6880752

In prior versions of Oracle Rdb, the ALTER INDEX ... MAINTENANCE IS ENABLED DEFERRED syntax was not active. Instead it produced a WISH_LIST error. This clause has now been activated for this release.

The following example shows the WISH_LIST error generated in prior versions.

```
SQL> alter index t1_idx maintenance is enabled deferred;
%RDB-E-NO_META_UPDATE, metadata update failed
-RDB-F-WISH_LIST, feature has not been implemented
```

When ALTER INDEX ... MAINTENANCE IS ENABLED DEFERRED is used on an index, it has the following actions:

- If the index is MAINTENANCE IS ENABLED IMMEDIATE mode, that index is changed to a build−pending index. The side effect of this change is that the table upon which the index is defined can now only be read. The index partitions may now be rebuilt using REBUILD PARTITION or a combination of TRUNCATE PARTITION and BUILD PARTITION clauses.
  A warning message is issued upon successful execution of the ALTER INDEX command so that the database administrator is aware that the index is incomplete and also that the table upon which the index is defined is write−disabled so that INSERT, UPDATE and DELETE statements cannot be used.

  ```
  SQL> alter index t1_idx maintenance is enabled deferred;
  %RDB-W-META_WARN, metadata successfully updated with the reported warning
  -RDMS-W-IDXBLDPEND, index in build pending state - maintenance is disabled
  SQL>
  ```

After each partition is rebuilt, a final ALTER INDEX ... MAINTENANCE IS ENABLED IMMEDIATE statement will be required to make the index active and allow updates to the table. This final step validates all partitions, rolls up partition cardinality information and clears the build−pending state.

- If the index is MAINTENANCE IS ENABLED DEFERRED mode then there are no changes made to the index and the ALTER INDEX statement quietly succeeds.
- If the index is MAINTENANCE IS DISABLED then an error is raised. This ALTER INDEX operation is not compatible with such indices. However, ALTER INDEX ... TRUNCATE ALL PARTITIONS or ALTER INDEX ... TRUNCATE PARTITION can be used to modify a disabled index into a build−pending index.

```
SQL> alter index t1_idx maintenance is enabled deferred;
%RDB−E−NO_META_UPDATE, metadata update failed
−RDMS−E−INDMAINTDIS, maintenance on index T1_IDX has been disabled
SQL>
```

This problem has been corrected in Oracle Rdb Release 7.2.3.

7.1.7 SQL Precompiler and Module Language Compiler /ARCHITECTURE Command Line Qualifier

For improved performance of generated code, the SQL Language Precompiler and Module Language Compilers include support for the /ARCHITECTURE=keyword command line qualifier on OpenVMS Alpha systems. At present, the /ARCHITECTURE qualifier is ignored on Itanium systems.

The /ARCHITECTURE=keyword qualifier specifies the lowest version of the Alpha architecture where this code will run which can allow the compiler to generate more efficient code with the tradeoff that code may not run on older systems.

All Alpha processors implement a core set of instructions and, in some cases, the following extensions:

- Byte/word extension (BWX) − The instructions that comprise the BWX extension are LDBU, LDWU, SEXTB, SEXTW, STB, and STW.
- Square−root and floating−point convert extension (FIX) − The instructions that comprise the FIX extension are FTOIS, FTOIT, ITOFF, ITOFS, ITOFT, SQRTF, SQRTG, SQRTS, and SQRTT.
- Count extension (CIX) − The instructions that comprise the CIX extension are CTLZ, CTPOP, and CTTZ.
- Multimedia extension (MVI) − The instructions that comprise the MVI extension are MAXSB8, MAXSW4, MAXUB8, MAXUW4, MINSB8, MINSW4, MINUB8, MINUW4, PERR, PKLB, PKWB, UNPKBL, and UNPKBW.

The Alpha Architecture Reference Manual describes the extensions in detail.

The keyword specified with the /ARCHITECTURE qualifier determines which instructions the compiler can generate and which coding rules it must follow.

- GENERIC − Generate instructions that are appropriate for all Alpha processors. This option is the default and is equivalent to /ARCH=EV4.
- HOST − Generate instructions for the processor that the compiler is running on (for example, EV56

instructions on an EV56 processor, EV7 instructions on an EV7 processor, and so on).

- EV4 − Generate instructions for the EV4 processor (21064, 20164A, 21066, and 21068 chips). Applications compiled with this option will not incur any emulation overhead on any Alpha processor.
- EV5 − Generate instructions for the EV5 processor (some 21164 chips). (Note that the EV5 and EV56 processors both have the same chip number − 21164.) Applications compiled with this option will not incur any emulation overhead on any Alpha processor.
- EV56 − Generate instructions for EV56 processors (some 21164 chips). This option permits the compiler to generate any EV4 instruction, plus any instructions contained in the BWX extension. Applications compiled with this option may incur emulation overhead on EV4 and EV5 processors.
- PCA56 − Generate instructions for PCA56 processors (21164PC chips). This option permits the compiler to generate any EV4 instruction plus any instructions contained in the BWX and MVI extensions. Applications compiled with this option may incur emulation overhead on EV4 and EV5 processors.
- EV6 − Generate instructions for EV6 processors (21264 chips). This option permits the compiler to generate any EV4 instruction, any instruction contained in the BWX and MVI extensions, plus any instructions added for the EV6 chip. These new instructions include a floating−point square root instruction (SQRT), integer/floating−point register transfer instructions, and additional instructions to identify extensions and processor groups. Applications compiled with this option may incur emulation overhead on EV4, EV5, EV56, and PCA56 processors.
- EV67 or EV68 − Generate instructions for EV67 and EV68 processors (21264A chips). This option permits the compiler to generate any EV6 instruction, plus the new bit count instructions (CTLZ, CTPOP, and CTTZ). However, the precompilers do not currently generate any of the new bit count instructions and the EV67 and EV68 have identical instruction scheduling models so the EV67 and EV68 are essentially identical to the EV6. Applications compiled with this option may incur emulation overhead on EV4, EV5, EV56, and PCA56 processors.
- EV7 − Generate instructions for the EV7 processor (21364 chip). This option permits the compiler to generate any EV67 instruction. There are no additional instructions available on the EV7 processor but the compiler does have different instruction scheduling and prefetch rules for tuning code for the EV7. Applications compiled with this option may incur emulation overhead on EV4, EV5, EV56, and PCA56 processors.

The OpenVMS Alpha operating system includes an instruction emulator. This capability allows any Alpha chip to execute and produce correct results from Alpha instructions − even if some of the instructions are not implemented on the chip. Applications using emulated instructions will run correctly but may incur significant emulation overhead at run time.

Of the available extension types, the Byte/word extension (BWX) will often be beneficial for increased performance of Rdb−based applications. In addition, for those Alpha implementations that support quad−issue of instructions (the EV6 and later processors), the compiler does have different instruction scheduling and prefetch rules for tuning code.

For highest levels of performance of generated code, Oracle recommends that the /ARCHITECTURE switch be specified with the keyword that most closely matches the lowest processor type of the machine where the program will execute.

Language Compiler Support for /ARCHITECTURE

If specified, the /ARCHITECTURE qualifier is passed on the command line to the specified language compiler by the SQL Precompiler. The language compiler being used must support the /ARCHITECTURE qualifier and the architecture keyword value when the

/ARCHITECTURE qualifier is specified.

7.1.8 PERFT4_RDB Example Program

Accessing performance information in a tabular fashion for Oracle Rdb databases can be often be beneficial. In particlar, data in "CSV" (comma separated values) format can be readily loaded into other applications. HP's T4 utility program TlViz expects data in an enhanced CSV format where the first 4 lines of the data file contain additional information.

SQL$SAMPLE:PERFT4_RDBxx.C is a sample C program that reads an RMU /SHOW STATISTICS binary file and converts all statistic values for each sample into a current rate per second. The statistics values are written to an output text file. This example is supplied as a template for writing your own program.

To use the PERFT4_RDB program, optionally compile and link (the example program has been supplied also as a .EXE) and then create a foreign command symbol with a value of "SQLSAMPLE:PERFT4_RDBxx.EXE" (where xx is the version of Rdb) and pass an input binary statistics file name and an output text file name. The following example command sequence demonstrates one possible way that statistics can be gathered for one hour and then formatted.

```
$ PERFT4 := $SQL$SAMPLE:PERFT4_RDB72
$ RMU /SHOW STATISTICS MFP −
        /NOINTERACTIVE −
        /OUTPUT = 2008−11−16−00−56.STATS −
        /UNTIL = "16−NOV−2008 11:00:00" −
        /TIME = 15
$ PERFT4 2008−11−16−00−56.STATS 2008−11−16−00−56.CSV
Wrote 251 records (from 16−Nov−2008 09:56:29 to 16−Nov−2008 10:59:57)
```

This example source code SQL$SAMPLE:PERFT4_RDBxx.C is intended solely to be used as a template for writing your own program. No support for this example template program is expressed or implied. Oracle Corporation assumes no responsibility for the functionality, correctness or use of this example program. Oracle Corporation reserves the right to change the format and contents of the Oracle Rdb RMU SHOW STATISTICS binary output file at any time without prior notice.

7.1.9 RMU Load Quietly Truncated String Data During Insert

Bug 6732438

In prior versions of Oracle Rdb, the RMU Load command would quietly truncate data loaded into CHAR and VARCHAR columns. This loss of data might be significant but was never reported by Oracle Rdb.

This problem has been corrected in Oracle Rdb Release 7.2.3. RMU Load now defaults to SQL dialect SQL99 which implicitly checks for and reports truncations during INSERT operations.

This behavior is controlled by a new /DIALECT qualifier.

- /NODIALECT, /DIALECT=SQL89 or /DIALECT=NONE will revert to prior behavior and no truncation error will be reported.
- /DIALECT=SQL99 (the default) will enable reporting of truncation errors. Note that truncation occurs if non−space characters are discarded during the insert.

```
$ rmu/load abc f2 f1
%RMU-I-LOADERR, Error loading row 1.
%RDB-E-TRUN_STORE, string truncated during assignment to a column
%RMU-I-DATRECREAD,  1 data records read from input file.
%RMU-I-DATRECSTO,   0 data records stored.
%RMU-F-FTL_LOAD, Fatal error for LOAD operation at 13-FEB-2008 15:39:44.40
$
```

7.1.10 New FETCH FIRST and OFFSET Clauses for Select Expression

This release of Oracle Rdb adds two new clauses to the SELECT expression. These clauses are defined by the draft SQL Database Language Standard 2008.

- OFFSET skip-expression
  The OFFSET clause allows the database programmer to start fetching the result rows from the specified offset within the result table. OFFSET accepts a numeric value expression which may contain arbitrary arithmetic operators, function calls, subselect clauses or sequence references. The subselect clauses may not reference columns in the outer query as it is evaluated before row processing begins.

  Note

  Oracle recommends that the values specified for skip-expression be kept small for performance reasons. The skipped rows are still fetched and processed by the query; they are just not returned to the application.

  The OFFSET clause is equivalent in functionality to the SKIP clause currently supported by the LIMIT TO clause. The distinction is that OFFSET can be specified without a row limit.
  OFFSET is not compatible with the SKIP (or OFFSET) sub clause of the LIMIT TO clause. However, OFFSET and LIMIT TO can be used together.
- FETCH FIRST limit-expression
- FETCH NEXT limit-expression
  The FETCH FIRST clause allows the database programmer to limit the results returned from a query expression. The FETCH FIRST clause is equivalent to functionality currently supported by the LIMIT TO clause. FETCH accepts a numeric value expression which may contain arbitrary arithmetic operators, function calls, subselect clauses or sequence references. The subselect clauses may not reference columns in the outer query as it is evaluated before row processing begins.
  The FETCH NEXT is identical to FETCH FIRST but allows the syntax to be more descriptive when coupled with the OFFSET clause.
  If no value expression is provided for FETCH, it will default to 1 row.
  The FETCH clause is not compatible with the LIMIT TO clause.

Examples

The following examples show the use of the FETCH FIRST and OFFSET clauses.

This example uses the DEPARTMENTS table to locate the employee id of each manager and, after sorting them by their birthday, the oldest manager's name and employee id are displayed.

Example 7−5 Using FETCH FIRST to find the oldest manager in the company

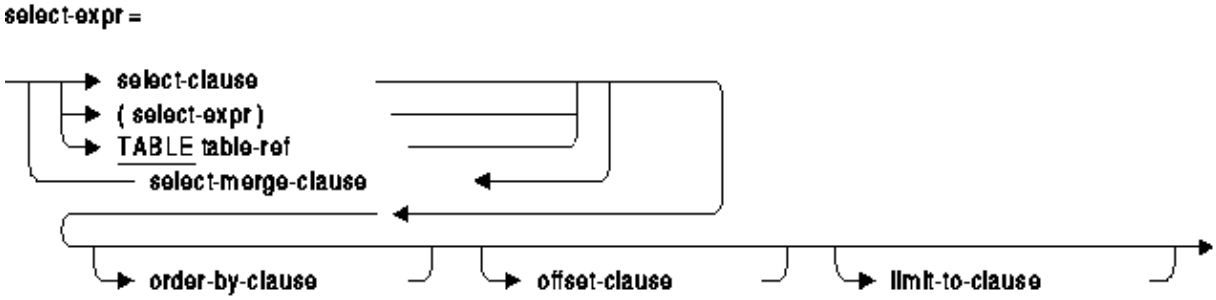
```
SQL> -- select the most senior manager
SQL> select e.last_name, e.first_name, e.employee_id
cont> from departments d, employees e
cont> where d.manager_id = e.employee_id
cont> order by e.birthday
cont> fetch first row only;
 E.LAST_NAME      E.FIRST_NAME   E.EMPLOYEE_ID
 O'Sullivan       Rick           00190
1 row selected
SQL>
```

This query uses a subselect in the OFFSET clause to locate the median (or middle) row of the sorted set.

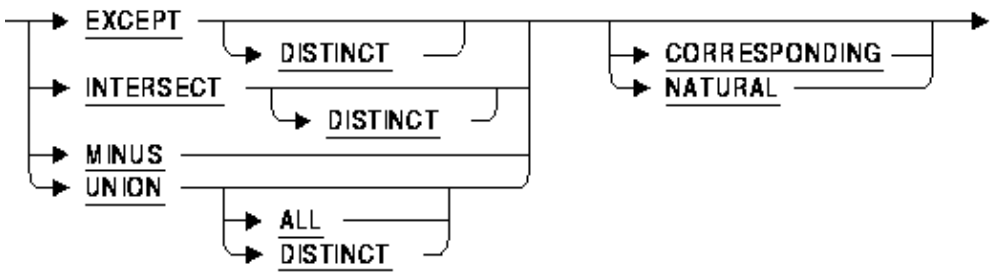
Example 7−6 Using OFFSET ROWS and FETCH NEXT to compute the median salaried employee

```
SQL> select e.last_name, e.first_name, employee_id, sh.salary_amount
cont> from salary_history sh inner join employees e using (employee_id)
cont> where sh.salary_end is null
cont> order by sh.salary_amount
cont> offset (select count(*)
cont>         from salary_history
cont>         where salary_end is null)/2 rows
cont> fetch next row only;
 E.LAST_NAME      E.FIRST_NAME   EMPLOYEE_ID   SH.SALARY_AMOUNT
 Boyd             Ann            00244               $24,166.00
1 row selected
SQL>
```

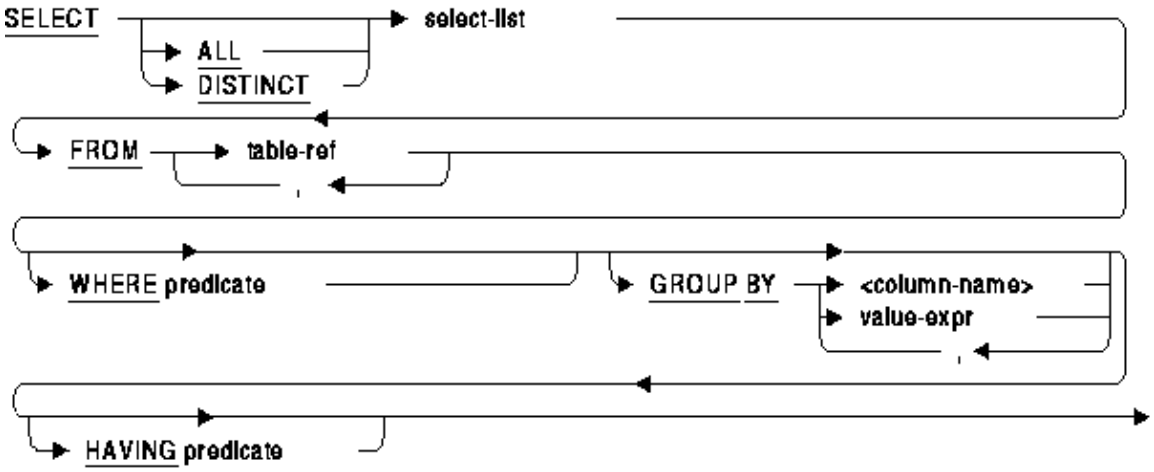
Syntax



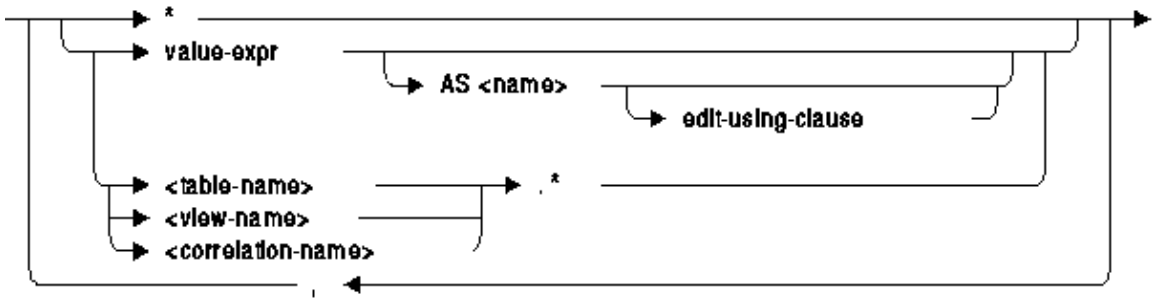
select-merge-clause =



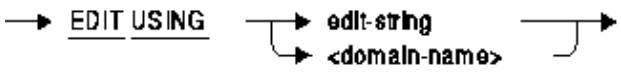
select-clause =



select-list =

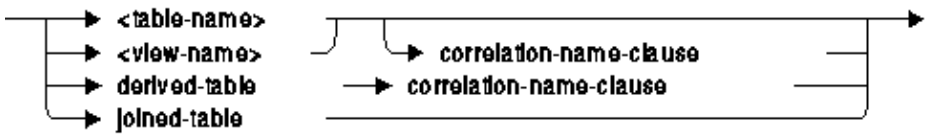


edit-using-clause =



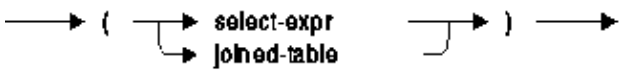
---

table-ref =



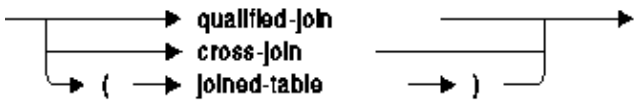
---

derived-table =



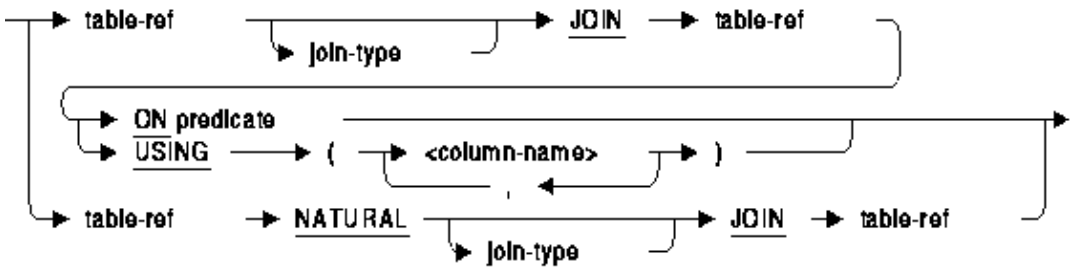
---

joined-table =



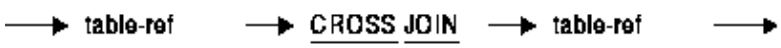
---

qualified-join =



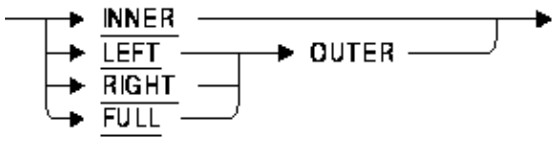
---

cross-join =



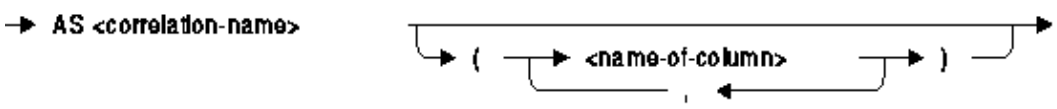
---

join-type =



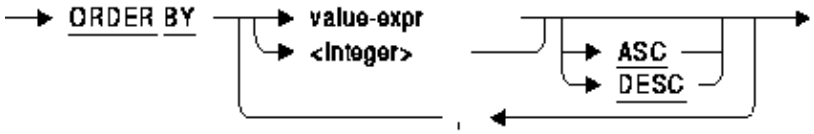
---

correlation-name-clause =



---

order-by-clause =

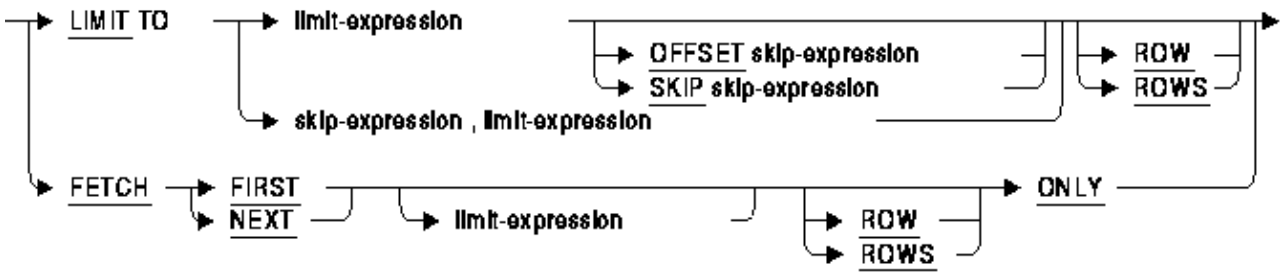


---

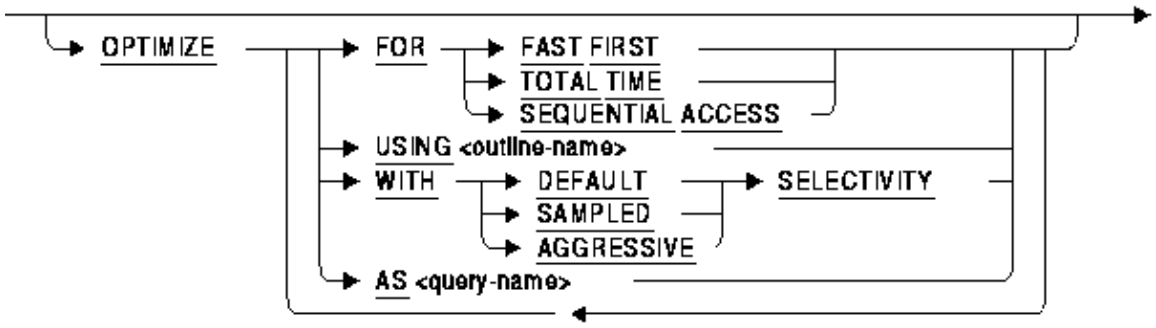
offset-clause =

---

limit-to-clause =



optimize-clause =



Usage Notes

- If ORDER BY is used in a query that includes OFFSET, FETCH FIRST (FETCH NEXT), or LIMIT TO clauses then the rows are first retrieved and sorted prior to applying the OFFSET, FETCH or LIMIT TO actions.
- A select expression may contain both OFFSET and FETCH NEXT (LIMIT TO) clauses, in which case the OFFSET is applied first and then the FETCH NEXT clause. For instance, if the query would normally return 100 rows, then an OFFSET 20 would skip over the first 20 rows and return the remaining 80. If on the other hand an OFFSET 20 and a LIMIT TO 20 were specified, then after skipping the first 20 rows the next 20 are returned.
- The OFFSET, FETCH FIRST or LIMIT TO clauses may result in no rows being retrieved.

7.1.11 RMU Unload Now Creates SQL*Loader Control Files

Enhancement 2146782

This release of Oracle Rdb has enhanced RMU Unload by providing support for SQL*Loader control files and portable data files.

The following example shows the FORMAT=CONTROL option.

```
$ RMU/UNLOAD/RECORD_DEFINITION=(FORMAT=CONTROL,FILE=EMP) -
    SQL$DATABASE -
    EMPLOYEES -
    EMPLOYEES
```

This command creates a file EMP.CTL (the SQL*Loader control file) and EMPLOYEES.DAT in a portable format to be loaded.

Usage Notes

- FORMAT=CONTROL implicitly uses a portable data format as TEXT rather than binary values. The unloaded data files are similar to that generated by FORMAT=TEXT but include a NULL vector to represent NULL values ('1') and non−NULL values ('0').
  The SQL*Loader control file uses this NULL vector to set NULL for the data upon loading.
- When FORMAT=CONTROL is used, the output control file and associated data file are intended to be used with the Oracle RDBMS SQL*Loader (sqlldr) command to load the data into an Oracle RDBMS database table. LIST OF BYTE VARYING (SEGMENTED STRING) columns are not unloaded.
  The file specification for the FILE option will default to a .CTL file type.
  Oracle Rdb does not support this format for RMU Load.
- The keywords NULL, PREFIX, SEPARATOR, SUFFIX, and TERMINATOR only apply to DELIMITED_TEXT format and may not be used in conjunction with the CONTROL keyword.
- DATE VMS data is unloaded including the fractional seconds precision. However, when mapped to Oracle DATE type in the control file, the fractional seconds value is ignored. It is possible to modify the generated control file to use the TIMESTAMP type and add FF to the date edit mask.

Chapter 8
Enhancements And Changes Provided in Oracle Rdb Release 7.2.2.0

8.1 Enhancements And Changes Provided in Oracle Rdb Release 7.2.2.0

8.1.1 Reduced Executable Image Sizes, Reduced CPU Usage, and Improved Performance

Several performance enhancements have been implemented in this release of Oracle Rdb. Most of these changes are either specific to applications running on I64 systems or will have a greater effect on I64 systems. These enhancements include:

- Streamlined code sequences
- Reduced alignment faults
- Reduction in executable image file size
- Enhancements to the optional run−time routine native compiler on I64

8.1.2 New SET FLAGS Keywords to Control Optimization Level

Bug 6389282

The optimization levels TOTAL TIME and FAST FIRST can be specified in the following ways:

- on the query itself using the OPTIMIZE FOR clause,
- within the query environment using the SET OPTIMIZATION LEVEL command,
- during application compile using SQL pre−compiler option /SQLOPTIONS=OPTIMIZATION_LEVEL, or the SQL Module Language /OPTIMIZATION_LEVEL qualifier,
- and specified via a query outline with the EXECUTION OPTIONS clause.

However, some dynamic SQL environments generate queries that cannot be affected by any of these methods. Therefore, Oracle Rdb has added a new flag to SET FLAGS (and RDMS$SET_FLAGS logical name) that can cover these types of queries.

The flag OPTIMIZATION_LEVEL can be used to change the default optimization level for a query. If the query explicitly uses the OPTIMIZE FOR clause or is compiled within an environment which overrides the default using the methods listed above, then no change will occur to the query optimization. If the query uses the default optimization level then their optimization will be modified by this flag.

- OPTIMIZATION_LEVEL with no option list (or an empty options list) will default to TOTAL TIME.
- NOOPTIMIZATION_LEVEL will revert to the default Oracle Rdb behavior.
- OPTIMIZATION_LEVEL(FAST_FIRST) will establish FAST FIRST as the default for queries in all sessions.
- OPTIMIZATION_LEVEL(TOTAL_TIME) will establish TOTAL TIME as the default for queries in all sessions.

- No other options are accepted for this keyword.

The following example shows the change of behavior for a query using the dynamic optimizer.

```
SQL> -- show with default behavior (FFirst tactic used)
SQL> select *
cont> from xtest
cont> where col2 between 999980 and 1000000
cont>   and col1 > 0
cont> ;
Tables:
  0 = XTEST
Leaf#01 FFirst 0:XTEST Card=10
  Bool: (0.COL2 >= 999980) AND (0.COL2 <= 1000000) AND (0.COL1 > 0)
  BgrNdx1 XTEST_IDX [1:0] Fan=17
    Keys: 0.COL1 > 0
0 rows selected
SQL>
SQL> -- use SET FLAGS
SQL> set flags 'optimization_level(total_time)';
SQL>
SQL> -- show that BgrOnly is used for TOTAL TIME
SQL> select *
cont> from xtest
cont> where col2 between 999980 and 1000000
cont>   and col1 > 0
cont> ;
Tables:
  0 = XTEST
Leaf#01 BgrOnly 0:XTEST Card=10
  Bool: (0.COL2 >= 999980) AND (0.COL2 <= 1000000) AND (0.COL1 > 0)
  BgrNdx1 XTEST_IDX [1:0] Fan=17
    Keys: 0.COL1 > 0
0 rows selected
SQL>
```

This feature has been added in Oracle Rdb Release 7.2.2.

8.1.3 New /ABMS_ONLY Qualifier to Only Dump Rdb Database ABM Pages

Currently, Oracle Rdb database Area Bit Map (ABM) pages can be dumped along with other types of pages for uniform storage areas or logical areas within uniform storage areas using the RMU/DUMP command. A new qualifier has been added to the RMU/DUMP command, /ABMS_ONLY, which will only dump ABM pages in uniform storage areas or in logical areas contained within uniform storage areas. The ABM pages can be dumped within a limited page range specified by the existing /START=n and/or /END=n qualifiers, where n is a page number; or if a limited page range is not specified, all ABM pages within uniform storage areas or within logical areas contained in uniform storage areas can be dumped.

The /ABMS_ONLY qualifier cannot be negated and will not be the default. The default if /ABMS_ONLY is not specified will continue to be to dump ABM pages along with other types of database pages for storage areas and logical areas. If the existing /AREAS or /LAREAS or /ALL_LIVE or /ALL_LAREA qualifiers are not used with the /ABMS_ONLY qualifier in the RMU/DUMP command line, a default of /ALL_LIVE is assumed to dump only ABM pages contained within all live uniform storage areas. The dump format for the ABM page has not changed. Headers will be output as they are currently to identify the live uniform storage

area being dumped and/or the logical area within the live uniform storage area followed by the ABM pages contained within each logical area. A dump of the database header will be included at the start of the dump in the same cases where it is currently included.

If there are no ABM pages within the specified page range or the storage area is a mixed format area or the logical area is contained within a mixed storage area, no ABM pages will be dumped since there are no ABM pages in these cases. If you execute the RMU/DUMP/HEADER command, the entries for storage areas defined for the Rdb database will specify which areas are of uniform format and which are of mixed format. The /ABMS_ONLY qualifier cannot be specified in the same dump command as the existing /SPAMS_ONLY qualifier which only dumps Space Management (SPAM) pages. The /ABMS_ONLY qualifier cannot be specified in the same dump command with the existing /SNAPSHOTS qualifier for dumping SNAPSHOT areas.

The syntax for dumping only ABM pages is as follows:

```
/ABMS_ONLY
```

In the following example, all ABM pages contained in all uniform storage areas in the specified Rdb database are dumped.

```
$ rmu/dump/abms_only/out=dmp.out mf_personnel
```

In the following example, only the ABM pages contained in the named uniform storage area in the specified Rdb database are dumped.

```
$ rmu/dump/abms_only/area=rdb$system mf_personnel
```

In the following example, only the ABM pages contained in the named logical area in a uniform storage area in the specified Rdb database are dumped.

```
$ rmu/dump/abms_only/larea=rdb$relations mf_personnel
```

In the following example, only the ABM pages contained within the specified page range in the named uniform storage area in the specified Rdb database are dumped.

```
rmu/dump/abms_only/area=rdb$system/start=1/end=5 mf_personnel
```

8.1.4 RMU/BACKUP Performance Enhancement

With storage areas located on different devices, an RMU/BACKUP would start reading all storage areas from the first input device before proceeding to the next input device. This caused an imbalance of I/O loads on the various devices. While one input device was highly active performing read I/Os, other input devices were idling.

As an enhancement in this release, RMU/BACKUP assigns storage areas to be saved to reader threads by using a round−robin scheme based on the disk devices of the storage areas. Larger storage areas for this disk device are selected first. As in the past, it still balances the amount of data that goes to each output device or save set or media−manager stream.

```
$ RMU/BACKUP/LOG=FULL LDA100:[DB]MYTESTDB −
    $1$DGA20:[BACKUP]RBF1,$1$DGA40:[BACKUP]RBF2/DISK=WRITERS=2
```

```
%RMU-I-BCKTXT_01, Writer thread 1 writes $1$DGA20:[BACKUP]RBF1.RBF; containing:
    File LDA100:[DB]MYTESTDB.RDA;1    1404 blocks
    File LDA100:[DB]A0007.RDA;1       5388 blocks
    File LDA300:[DB]A0006.RDA;1       7758 blocks
    File LDA100:[DB]A0004.RDA;1       4490 blocks
    File LDA200:[DB]A0005.RDA;1       9310 blocks
    File LDA300:[DB]A0009.RDA;1       4490 blocks
    File LDA200:[DB]A0011.RDA;1       3742 blocks
    File LDA100:[DB]A0001.RDA;1       2599 blocks
    File LDA300:[DB]A0012.RDA;1       3742 blocks
%RMU-I-BCKTXT_01, Writer thread 2 writes $1$DGA40:[BACKUP]RBF2.RBF; containing:
    File LDA200:[DB]A0002.RDA;1      27802 blocks
    File LDA100:[DB]A0010.RDA;1       3742 blocks
    File LDA300:[DB]A0003.RDA;1       4490 blocks
    File LDA200:[DB]A0008.RDA;1       2166 blocks
%RMU-I-BCKTXT_00, Backed up root file LDA100:[DB]MYTESTDB.RDB;1
%RMU-I-RESUME, resuming operation on volume 2 using _$1$DGA40
%RMU-I-BCKTXT_02, Starting full backup of storage area (RDB$SYSTEM)
LDA100:[DB]MYTESTDB.RDA;1
%RMU-I-BCKTXT_12, Completed full backup of storage area (RDB$SYSTEM)
LDA100:[DB]MYTESTDB.RDA;1
%RMU-I-BCKTXT_02, Starting full backup of storage area (A0009)
LDA300:[DB]A0009.RDA;1
%RMU-I-BCKTXT_02, Starting full backup of storage area (A0002)
LDA200:[DB]A0002.RDA;1
%RMU-I-BCKTXT_02, Starting full backup of storage area (A0007)
LDA100:[DB]A0007.RDA;1
%RMU-I-BCKTXT_02, Starting full backup of storage area (A0010)
LDA100:[DB]A0010.RDA;1
%RMU-I-BCKTXT_02, Starting full backup of storage area (A0006)
LDA300:[DB]A0006.RDA;1
%RMU-I-BCKTXT_02, Starting full backup of storage area (A0003)
LDA300:[DB]A0003.RDA;1
%RMU-I-BCKTXT_02, Starting full backup of storage area (A0004)
LDA100:[DB]A0004.RDA;1
%RMU-I-BCKTXT_02, Starting full backup of storage area (A0008)
LDA200:[DB]A0008.RDA;1
%RMU-I-BCKTXT_02, Starting full backup of storage area (A0005)
LDA200:[DB]A0005.RDA;1
%RMU-I-BCKTXT_12, Completed full backup of storage area (A0008)
LDA200:[DB]A0008.RDA;1
%RMU-I-BCKTXT_12, Completed full backup of storage area (A0010)
LDA100:[DB]A0010.RDA;1
%RMU-I-BCKTXT_12, Completed full backup of storage area (A0003)
LDA300:[DB]A0003.RDA;1
%RMU-I-BCKTXT_12, Completed full backup of storage area (A0004)
LDA100:[DB]A0004.RDA;1
%RMU-I-BCKTXT_02, Starting full backup of storage area (A0011)
LDA200:[DB]A0011.RDA;1
%RMU-I-BCKTXT_12, Completed full backup of storage area (A0009)
LDA300:[DB]A0009.RDA;1
%RMU-I-BCKTXT_02, Starting full backup of storage area (A0001)
LDA100:[DB]A0001.RDA;1
%RMU-I-BCKTXT_12, Completed full backup of storage area (A0007)
LDA100:[DB]A0007.RDA;1
%RMU-I-BCKTXT_02, Starting full backup of storage area (A0012)
LDA300:[DB]A0012.RDA;1
%RMU-I-BCKTXT_12, Completed full backup of storage area (A0001)
LDA100:[DB]A0001.RDA;1
%RMU-I-BCKTXT_12, Completed full backup of storage area (A0002)
LDA200:[DB]A0002.RDA;1
%RMU-I-BCKTXT_12, Completed full backup of storage area (A0006)
```

```
LDA300:[DB]A0006.RDA;1
%RMU-I-BCKTXT_12, Completed full backup of storage area (A0011)
LDA200:[DB]A0011.RDA;1
%RMU-I-BCKTXT_12, Completed full backup of storage area (A0005)
LDA200:[DB]A0005.RDA;1
%RMU-I-BCKTXT_12, Completed full backup of storage area (A0012)
LDA300:[DB]A0012.RDA;1
%RMU-I-COMPLETED, BACKUP operation completed
```

8.1.5 /NOOUTPUT Can Now Be Specified With the RMU/SET SERVER Command

The "RMU /SET SERVER /OUTPUT=filespec servertype" command can be used to specify the default output file specification for several of the database server processes. If the output file specification is empty, the output file entry is disabled. Now, in addition to specifying an empty output file specification with the /OUTPUT qualifier in order to disable the output file entry, "RMU /SET SERVER /NOOUTPUT servertype" can be specified as a way to disable the output file entry. Note that /NOOUTPUT is the default and if /OUTPUT is not specified the output file server logging entry will be disabled.

The syntax for the /OUTPUT qualifier is therefore the following.

```
[NO]OUTPUT[=filespec]
```

The following example shows that now the output file entry can be disabled by the RMU/SET SERVER command by specifing /NOOUTPUT or by specifying /OUTPUT with an empty output file specification or by not specifying /OUTPUT since /NOOUTPUT is the default.

```
$ RMU /SET SERVER LRS /NOOUTPUT DUA0:[ZDB]ZDB.RDB
$ RMU /SET SERVER LRS /OUTPUT="" DUA0:[ZDB]ZDB.RDB
$ RMU /SET SERVER LRS DUA0:[ZDB]ZDB.RDB
```

8.1.6 RMU /RESTORE Allows Change of Page Size For Uniform Format Storage Areas

Bug 705542

Previously, the RMU/RESTORE command allowed increasing the page size only for mixed−format storage areas.

This restriction has been relaxed. Page size may now be increased for both mixed and uniform storage area formats during an RMU restore operation.

Chapter 9
Enhancements And Changes Provided in Oracle Rdb Release 7.2.1.4

9.1 Enhancements And Changes Provided in Oracle Rdb Release 7.2.1.4

9.1.1 RMU/SHOW STATISTICS /STALL_LOG And /ALARM

Enhancement Bug 6331702

In previous Oracle Rdb releases, using the RMU/SHOW STATISTICS /STALL_LOG feature resulted in every active stall message being written to the stall log file at each sample interval. In some cases, this is an excessive amount of information as a result of most of the active stalls being of short duration.

This problem has been corrected in Oracle Rdb Release 7.2.1.4. The RMU/SHOW STATISTICS command now accepts a new LOG_STALL_ALARM keyword for the /OPTION qualifier. If LOG_STALL_ALARM is present when using /STALL_LOG and /ALARM, only those stalls exceeding the /ALARM specified duration are written to the stall log output file.

9.1.2 RMU/SHOW STATISTICS Stall Alarm Invoked Procedure Parameter Addition

Enhancement Bug 6331702

Parameter P6 has been added to the parameters passed to the stall alarm invoked command. The P6 parameter contains the formatting stall message string.

The parameters passed to the invoked command are the following:

Table 9−1 Stall Alarm Invoked Procedure Parameters

Parameter	Description
P1	Root file specification
P2	Current data/time
P3	Process ID
P4	Stream ID
P5	Alarm threshold seconds
P6	Formatted stall message

9.1.3 RMU /SHOW AIP New Qualifier /BRIEF

Enhancement Bug 3390639

The RMU /SHOW AIP command now supports the qualifier /BRIEF to display AIP information in a condensed, tabular form as in the following example:

```
$ RMU/SHOW AIP  DKA0:[DB]DB /PAREA=(4,5)/BRIEF
*-------------------------------------------------------------------------------
* Logical Area Name             LArea PArea   Len Type
*-------------------------------------------------------------------------------
RDB$SYSTEM_RECORD                  60     4   215 SYSTEM RECORD
RDB$SYSTEM_RECORD                  61     5   215 SYSTEM RECORD
EMPLOYEES_HASH                     79     4   215 HASH INDEX
EMPLOYEES                          82     4   121 TABLE
JOB_HISTORY_HASH                   85     4   215 HASH INDEX
JOB_HISTORY                        88     4    42 TABLE
DEPARTMENTS_INDEX                  89     5   430 SORTED INDEX
DEPARTMENTS                        90     5    55 TABLE
```

The columns displayed include:

- Logical Area Name − Name of the logical area stored in the AIP entry
- LArea − Logical area number stored in the AIP entry
- PArea − Physical area number stored in the AIP entry
- Len − Object length stored in the AIP entry
- Type − Object type stored in the AIP entry. The following object types may be displayed:
    - ♦ UNKNOWN − The logical area type is unknown or has not been set
    - ♦ TABLE − A data table type
    - ♦ SORTED INDEX − A sorted index type
    - ♦ HASH INDEX − A hashed index type
    - ♦ SYSTEM RECORD − A system record type
    - ♦ LARGE OBJECT − A large object (BLOB) type

9.1.4 RMU /SHOW AIP New Qualifier /PAREA

The RMU /SHOW AIP command now supports the qualifier /PAREA to display AIP information for logical areas stored in the specified physical areas.

```
$ RMU/SHOW AIP DKA0:[DB]DB /BRIEF /PAREA=(4,5)
*-------------------------------------------------------------------------------
* Logical Area Name             LArea PArea   Len Type
*-------------------------------------------------------------------------------
RDB$SYSTEM_RECORD                  60     4   215 SYSTEM RECORD
RDB$SYSTEM_RECORD                  61     5   215 SYSTEM RECORD
EMPLOYEES_HASH                     79     4   215 HASH INDEX
EMPLOYEES                          82     4   121 TABLE
JOB_HISTORY_HASH                   85     4   215 HASH INDEX
JOB_HISTORY                        88     4    42 TABLE
DEPARTMENTS_INDEX                  89     5   430 SORTED INDEX
DEPARTMENTS                        90     5    55 TABLE
```

9.1.5 New Options for RMU DUMP EXPORT Command

This release of Oracle Rdb adds new keywords to the OPTIONS qualifier for RMU Dump Export. The OPTIONS qualifier allows the user to modify the output from this dump command.

- ALLOCATION
  When importing databases for testing, the full allocation recorded in the interchange file is often not

required. The clauses ALLOCATION and SNAPSHOT ALLOCATION are controlled by this option. The default is ALLOCATION. Use NOALLOCATION to omit these clauses from the generated SQL script. This option is ignored if NOIMPORT_DATABASE is specified or defaulted for the OPTIONS qualifier.

- FILENAME_ONLY
  When importing databases for testing, the full file specification for the database root, storage areas and snapshot areas recorded in the interchange file is often not required. The FILENAME clauses are controlled by this option which trims the specification to only the filename portion. The default is NOFILENAME_ONLY. Use FILENAME_ONLY to truncate the file specification in generated SQL script. This option is ignored if NOIMPORT_DATABASE is specified or defaulted for the OPTIONS qualifier.
- HEADER_SECTION
  This option allows the database administrator to display just the header portion of the interchange file and avoid dumping the data or metadata for every row in the table.

```
$ RMU/DUMP/EXPORT/OPTION=HEADER JOBS.UNL

BEGIN HEADER SECTION - (0)
    NONCORE_TEXT HDR_BRP_ID - (20) :  Load/Unload utility
    CORE_NUMERIC HDR_BRPFILE_VERSION - (1) : 4
    NONCORE_TEXT HDR_DBS_ID - (18) : Oracle Rdb V7.2-10
    NONCORE_TEXT HDR_DB_NAME - (16) : DB$:MF_PERSONNEL
    NONCORE_DATE HDR_DB_LOG_BACKUP_DATE - (8) :  3-JUL-2006 16:52:32.83
    CORE_NUMERIC HDR_DATA_COMPRESSION - (1) : 1
END HEADER SECTION - (0)

$
```

  In this example, the output describes the creator of the interchange file (RMU/UNLOAD), the version of Rdb used to create the file, the file specification of the database used, the date and time the interchange file was created, and an indication that compression was used by RMU Unload.
- IMPORT_DATABASE
  This keyword requests that the output from RMU Dump Export be formatted as a SQL IMPORT DATABASE statement. It uses the database attributes present in the interchange file formatted as SQL clauses. Of particular interest is the CREATE STORAGE AREA clauses which are required to IMPORT the source interchange (.rbr) file.
  The keyword HEADER_SECTION is implicitly selected when IMPORT_DATABASE is used, limiting the I/O to the interchange file to the section containing the database attributes.
  The default is NOIMPORT_DATABASE.

Usage Notes

- If the source interchange file is created by RMU Unload, then it does not contain any IMPORT DATABASE information and the generated SQL script cannot be used to create a database from such an interchange file.

```
$ RMU/DUMP/EXPORT/OP=IMPORT_DATABASE EMPLOYEES.UNL/OUT=EMP.SQL
$ SQL$ @EMP.SQL
SQL> IMPORT DATABASE
cont>     from 'DISK1:[TESTING]EMPLOYEES.UNL;1'
cont>     -- ident ' Load/Unload utility'
cont>     -- backup file version 4
cont>     -- database ident 'Oracle Rdb V7.2-131'
```

```
cont>     filename 'DB$:MF_PERSONNEL'
cont> ;
%SQL-F-EXTRADATA, unexpected data at the end of the .RBR file
$
```

- The IMPORT_DATABASE option is intended to create a SQL script as an aid to the database administrator. Some editing of the generated script may be required under some circumstances. Only a subset of the database attributes are dumped by RMU for the IMPORT_DATABASE output. Continue to use the RMU Dump Export Option=NOIMPORT_DATABASE to see all attributes recorded in the interchange file.

9.1.6 RMU/UNLOAD/AFTER_JOURNAL /QUICK_SORT_LIMIT Qualifier

The RMU/UNLOAD/AFTER_JOURNAL performs a sort operation to eliminate duplicate record modifications for each transaction being extracted. For smaller sort cardinalities, an internal in−memory "quick sort" algorithm is used, otherwise the SORT32 algorithm is used. Previously, the limit for using the quick sort routine was a fixed value of 5000 records.

This restriction of a fixed value for the threshold has been relaxed in Oracle Rdb Release 7.2.1.4. A new qualifier /QUICK_SORT_LIMIT=n has been provided to allow explicitly controlling the maximum number of records that will be sorted with the in−memory algorithm. The default value is 5000. The minimum value is 10 and the maximum value is 100,000.

Larger values specified for the /QUICK_SORT_LIMIT qualifier may reduce sort work file IO at the expense of additional CPU time and/or memory consumption. A too small value may result in additional disk file IO. Oracle believes that, in general, the default value should be accepted.

Chapter 10
Enhancements And Changes Provided in Oracle Rdb Release 7.2.1.3

10.1 Enhancements And Changes Provided in Oracle Rdb Release 7.2.1.3

10.1.1 RMU /ANALYZE /INDEX Wildcard Support

The RMU /ANALYZE /INDEX command now processes the wildcard characters "%" and "*" in index name specifications.

The following examples demonstrate various combinations of use of the wildcard patterns:

```
$ RMU /ANALYZE /INDEX MF_PERSONNEL EMP*
$ RMU /ANALYZE /INDEX MF_PERSONNEL *LAST%NAME
$ RMU /ANALYZE /INDEX MF_PERSONNEL EMP%LAST%NAME
$ RMU /ANALYZE /INDEX MF_PERSONNEL *HASH, *LAST*
```

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

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

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

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

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

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

```
$ RMU/VERIFY/ALL DISK:[DIRECTORY]MF_PERSONNEL
%RMU−E−BADCLTSEQALLOC, 32 client sequences allocated in the root is less
 than 55 client sequences defined in RDB$SEQUENCES.
%RMU−E−BADCLTSEQMAXID, 32 client sequences allocated in the root is less
 than the maximum client sequence id of 55 in RDB$SEQUENCES.
%RMU−E−NOSEQENT, sequence id 33 has no valid entry in the root file
```

```
%RMU-E-NOSEQENT, sequence id 34 has no valid entry in the root file
%RMU-E-NOSEQENT, sequence id 35 has no valid entry in the root file
%RMU-E-NOSEQENT, sequence id 36 has no valid entry in the root file
%RMU-E-NOSEQENT, sequence id 37 has no valid entry in the root file
%RMU-E-NOSEQENT, sequence id 38 has no valid entry in the root file
%RMU-E-NOSEQENT, sequence id 39 has no valid entry in the root file
%RMU-E-NOSEQENT, sequence id 40 has no valid entry in the root file
%RMU-E-NOSEQENT, sequence id 41 has no valid entry in the root file
%RMU-E-NOSEQENT, sequence id 42 has no valid entry in the root file
%RMU-E-NOSEQENT, sequence id 43 has no valid entry in the root file
%RMU-E-NOSEQENT, sequence id 44 has no valid entry in the root file
%RMU-E-NOSEQENT, sequence id 45 has no valid entry in the root file
%RMU-E-NOSEQENT, sequence id 46 has no valid entry in the root file
%RMU-E-NOSEQENT, sequence id 47 has no valid entry in the root file
%RMU-E-NOSEQENT, sequence id 48 has no valid entry in the root file
%RMU-E-NOSEQENT, sequence id 49 has no valid entry in the root file
%RMU-E-NOSEQENT, sequence id 50 has no valid entry in the root file
%RMU-E-NOSEQENT, sequence id 51 has no valid entry in the root file
%RMU-E-NOSEQENT, sequence id 52 has no valid entry in the root file
%RMU-E-NOSEQENT, sequence id 53 has no valid entry in the root file
%RMU-E-NOSEQENT, sequence id 54 has no valid entry in the root file
%RMU-E-NOSEQENT, sequence id 55 has no valid entry in the root file
%RMU-E-BADCLTSEQUSED, 32 client sequences in use in the root does not
 equal 55 client sequences defined in RDB$SEQUENCES.
```

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

10.1.3 RMU/SHOW LOCKS Per Database New Feature

Enhancement Bug 6004181

In previous Rdb releases, using the RMU/SHOW LOCKS command could be difficult on systems with multiple open databases due to the amount of output and difficulty in determining what database a particular lock references.

This problem has been corrected in Oracle Rdb Release 7.2.1.3. The RMU/SHOW LOCKS command now accepts a root file specification that can be used in some cases to additionally filter lock displays to a specific database.

Note that in some cases the RMU/SHOW LOCKS command may be unable to filter locks prior to display. And when using the database "LOCK PARTITIONING IS ENABLED" feature for a database, the RMU/SHOW LOCKS command with a root file specification will be unable to associate area, page, and record locks with the specified database because the database lock is not the lock tree root for these lock types.

10.1.4 Reduced CPU Usage and Improved Performance

Several performance enhancements have been implemented in this release of Oracle Rdb, Release 7.2.1.3. Most of these changes are either specific to applications running on I64 systems or will have a greater effect on I64 systems. These enhancements include:

- Streamlined code sequences
- Reduced alignment faults

- Enhancements to the Optional Run−Time Routine Native Compiler on I64

10.1.5 Sample of Rdb External Routine Access to Oracle RDBMS

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

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

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

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

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

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

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

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

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

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

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

- roif_close_emp_cursor − a function
- roif_update_employee − a procedure
- roif_insert_employee − a procedure

10.1.6 New RMU /SET DATABASE /TRANSACTION_MODE=(...) Command

Bug 6047140

A new RMU /SET command "DATABASE /TRANSACTION_MODE=(...)" has been added to allow altering of the database allowed transaction modes without marking the database as modified. This command is intended to be used to set the transaction modes allowed on a standby database. This command requires exclusive database access (the database cannot be open or be accessed by other users).

Because only read−only transactions are allowed on a standby database, you may wish to use the TRANSACTION_MODE=READ_ONLY qualifier setting on a standby database. This setting prevents modifications to the standby database at all times, even when replication operations are not active.

The RMU /SET DATABASE command requires a database specification. Valid keywords for the "RMU /SET DATABASE /TRANSACTION_MODE=(...)" qualifier are:

- ALL − Enables all transaction modes
- CURRENT − Enables all transaction modes that are set in the database
- NONE − Disables all transaction modes
- [NO]BATCH_UPDATE
- [NO]READ_ONLY
- [NO]EXCLUSIVE
- [NO]EXCLUSIVE_READ
- [NO]EXCLUSIVE_WRITE
- [NO]PROTECTED
- [NO]PROTECTED_READ
- [NO]PROTECTED_WRITE
- [NO]READ_WRITE
- [NO]SHARED
- [NO]SHARED_READ
- [NO]SHARED_WRITE

If you specify more than one transaction mode in the mode−list, enclose the list in parentheses and separate the transaction modes from one another with a comma. Note the following:

- When you specify a negated transaction mode, it indicates that a mode is not an allowable access mode. For example, if you specify the Noexclusive_Write access mode, it indicates that exclusive write is not an allowable access mode for the restored database.
- If you specify the Shared, Exclusive, or Protected transaction mode, Oracle RMU assumes you are referring to both reading and writing in that transaction mode.
- No mode is enabled unless you add that mode to the list, or you use the All option to enable all transaction modes.
- You can list one transaction mode that enables or disables a particular mode followed by another that does the opposite.

For example, /TRANSACTION_MODE=(NOSHARED_WRITE, SHARED) is ambiguous because the first value disables Shared_Write access and the second value enables Shared_Write access. Oracle RMU resolves the ambiguity by first enabling the modes as specified in the modes−list and then disabling the modes as specified in the modes−list. The order of items in the list is irrelevant. In the example presented previously, Shared_Read is enabled and Shared_Write is disabled.

10.1.7 COMPRESS Qualifier for After−Image Journal Backup Command

Bug 6009124

After−Image Journal backup files can now be compressed the same way database backup files can be compressed. Compression for AIJ backup files can be combined with encryption. See HELP for the usage of the COMPRESSION and ENCRYPT qualifier for RMU backup.

The following commands have been modified to work with compressed AIJ backup files:

```
RMU /BACKUP /AFTER_JOURNAL /COMPRESSION
RMU /DUMP   /AFTER_JOURNAL
RMU /RECOVER
```

Compression is only supported for AIJ backup files using the NEW_TAPE format. Therefore all commands listed above must have /FORMAT=NEW_TAPE added to the command line.

The /LOG qualifier added to the RMU /BACKUP /AFTER_JOURNAL command reports the achieved compression at the end of the log output.

```
RMU /BACKUP /AFTER /FORMAT=NEW_TAPE  FOO.RDB FOO.ABF /COMPRESS=ZLIB /LOG
%RMU−I−AIJBCKBEG, beginning after−image journal backup operation
...
%RMU−I−LOGCOMPR, data compressed by 55% (27152 KB in/12471 KB out)
```

10.1.8 New Qualifier /[NO]DATABASE_VERIFICATION For RMU /BACKUP Command

Enhancement Bug 4940557

The RMU /BACKUP command performs a limited database root file verification at the start of the backup operation. This verification is intended to help prevent backing up a database with various detectable corruptions or inconsistancies of the root file or associated database structures. However, in some limited cases, it can be desirable to avoid these checks.

A new qualifier /[NO]DATABASE_VERIFICATION has been added to the RMU /BACKUP command. The default behavior is /DATABASE_VERIFICATION. /NODATABASE_VERIFICATION may be specified to avoid the database root file verification at the start of the backup. Oracle strongly recommends accepting the default of /DATABASE_VERIFICATION.

10.1.9 Qualifier /[NO]CONFIRM For RMU /RECOVER Command

The /CONFIRM qualifier for the RMU /RECOVER command causes the operator to be queried when an incorrect sequence of AIJ files is detected.

In the following example, note that the backed up AIJ files are specified in the order B1, B3, B2, B4 representing sequence numbers 1, 3, 2, 4:

```
$ RMU/RECOVER/NOLOG B1,B3,B2,B4
%RMU-I-LOGRECDB, recovering database file $1$DGA203:[DB]FOO.RDB;1
%RMU-W-AIJSEQPRI, AIJ file sequence number 1 created prior to
expected sequence 2
%RMU-I-LOGRECSTAT, transaction with TSN 0:224 ignored
%RMU-I-AIJONEDONE, AIJ file sequence 1 roll-forward operations completed
%RMU-W-NOTRANAPP, no transactions in this journal were applied
%RMU-W-AIJSEQAFT, incorrect AIJ file sequence 3 when 2 was expected
Do you wish to continue the roll-forward operation [N]:
```

RMU detects the improper journal order and displays the message "RMU−W−AIJSEQAFT, incorrect AIJ file sequence 3 when 2 was expected". RMU then asks the operator if the roll−forward operation using the incorrect AIJ file sequence 3 should be allowed to continue. If the operator specifies "Y", then the roll−forward operation on AIJ file sequence 3 will continue. Otherwise, RMU will move to the next journal (AIJ file sequence 2 in this example).

Note

Oracle recommends that, in general, an incorrect journal sequence not be applied as a corrupt database may result.

The /ORDER_AIJ_FILES qualifier can be used to help ensure that the specified journals are applied in the correct order.

The default setting for the /CONFIRM qualifier is /NOCONFIRM for batch processes and /CONFIRM otherwise.

10.1.10 /ORDER_AIJ_FILES Removes Some Unnecessary Files For RMU /RECOVER Command

The /ORDER_AIJ_FILES qualifier, in addition to ordering the specified input AIJ files by ascending sequence number, now also can eliminate some AIJ files from processing if they are known to be prior to the database recovery sequence starting point.

In the following example, note that the backed up AIJ files are specified in the order B1, B3, B2, B4 representing sequence numbers 1, 3, 2, 4. The /ORDER_AIJ_FILES sorts the journals to be applied into ascending sequence order and then is able to remove B1 from processing because the database recovery starts with AIJ file sequence 2 as shown in the RMU/RESTORE output.

```
$ RMU/RESTORE/NEW/NOCDD/NOAFTER FOO
%RMU-I-RESTXT_00, Restored root file DUA0:[DB]FOO.RDB;16
    .
```

```
    .
    .
%RMU-I-AIJRECFUL, Recovery of the entire database starts with
AIJ file sequence 2
%RMU-I-COMPLETED, RESTORE operation completed at 24-MAY-2007 12:23:32.99
$!
$ RMU/RECOVER/LOG/ORDER_AIJ_FILES B1,B3,B2,B4
    .
    .
    .
%RMU-I-LOGOPNAIJ, opened journal file DUA0:[DB]B2.AIJ;24
%RMU-I-LOGRECSTAT, transaction with TSN 0:256 ignored
%RMU-I-LOGRECSTAT, transaction with TSN 0:257 ignored
%RMU-I-RESTART, restarted recovery after ignoring 2 committed transactions
%RMU-I-AIJONEDONE, AIJ file sequence 2 roll-forward operations completed
%RMU-I-LOGRECOVR, 0 transactions committed
%RMU-I-LOGRECOVR, 0 transactions rolled back
%RMU-I-LOGRECOVR, 2 transactions ignored
%RMU-I-AIJNOACTIVE, there are no active transactions
%RMU-I-AIJSUCCES, database recovery completed successfully
%RMU-I-AIJNXTSEQ, to continue this AIJ file recovery, the
sequence number needed will be 3
    .
    .
    .
```

Note that due to the fact the AIJ backup files might have more than one journal sequence in them, it is not always possible for RMU to eliminate every journal file that might otherwise appear to be unneeded. But for those journals where RMU is able to know for certain that the journal will not be needed based on the database recovery restart information, journals can be avoided from having to be processed.

Chapter 11
Enhancements And Changes Provided in Oracle Rdb Release 7.2.1.2

11.1 Enhancements And Changes Provided in Oracle Rdb Release 7.2.1.2

11.1.1 Optional Run−Time Routine Native Compiler on I64

On Alpha and VAX systems, Oracle Rdb generates, at run time, callable subroutines of native machine instructions. These subroutines are executed during the processing of database insert/update/retrieval operations.

Oracle Rdb on I64 systems uses a hardware−portable instruction interpreter. This allowed rapid development and deployment of Oracle Rdb on Integrity OpenVMS systems. This release of Rdb introduces a second pass optimization that converts some of these portable instructions to native I64 machine code for enhanced performance.

For most applications, there is no significant performance reduction due to interpreted subroutines on I64. However, for some applications that process many records per compiled request and do relatively few IO operations, considerable CPU time may be spent interpreting these subroutines on I64 systems.

In order to improve performance for these applications, Oracle Rdb now includes an optional run−time compiler that is able to translate many generated subroutines into native Itanium instruction subroutines. This feature is optional and Oracle believes that most customers will not need to enable it. The only way to determine if this feature is a performance gain for your application is to test both with and without the feature enabled.

There is CPU overhead required to compile a subroutine and not all subroutines can be compiled, at present, into native Itanium routines. Additional virtual memory is required for the functionality of the optional run−time compiler.

In the future, Oracle anticipates extending this functionality to handle all possible generated subroutines. Oracle also expects that in the future, performance will continue to improve as this functionality is extended.

In order to enable this optional run−time compiler on I64 systems, define the logical name RDMS$BIND_CODE_OPTIMIZATION to a value of "2" or use the flag "CODE_OPTIMIZATION(2)". To disable the optional run−time compiler, either deassign the logical name or define it to a value of "0" or use the flag "CODE_OPTIMIZATION(0)". This logical name and flag have no effect on Alpha systems. The following examples show use of the logical name and the flag to enable and disable and show the status of the optional run−time compiler on I64 systems:

```
! To enable:
$ DEFINE RDMS$BIND_CODE_OPTIMIZATION 2
$ DEFINE RDMS$SET_FLAGS "CODE_OPTIMIZATION(2)"
SQL> SET FLAGS 'CODE_OPTIMIZATION(2)';

! To disable:
$ DEFINE RDMS$BIND_CODE_OPTIMIZATION 0
$ DEFINE RDMS$SET_FLAGS "CODE_OPTIMIZATION(0)"
SQL> SET FLAGS 'CODE_OPTIMIZATION(0)';

! Show current setting enabled and disabled:
SQL> ATT 'FI MF_PERSONNEL';
```

```
SQL> SET FLAGS 'CODE_OPTIMIZATION(2)';
SQL> SHOW FLAGS

Alias RDB$DBHANDLE:
Flags currently set for Oracle Rdb:
   PREFIX,WARN_DDL,INDEX_COLUMN_GROUP,MAX_SOLUTION
   ,MAX_RECURSION(100),REFINE_ESTIMATES(127)
   ,CODE_OPTIMIZATION(2),NOBITMAPPED_SCAN

SQL> SET FLAGS 'CODE_OPTIMIZATION(0)';
SQL> SHOW FLAGS

Alias RDB$DBHANDLE:
Flags currently set for Oracle Rdb:
   PREFIX,WARN_DDL,INDEX_COLUMN_GROUP,MAX_SOLUTION
   ,MAX_RECURSION(100),REFINE_ESTIMATES(127),NOBITMAPPED_SCAN
SQL>
```

If you detect a difference in the functionality or behavior of Oracle Rdb when this feature is enabled, you may deassign the logical name to revert to the prior (interpreted) execution model.

11.1.2 Mixed Case Passwords Supported

Bug 5916102

In prior versions of Oracle Rdb, attempts to use USER and USING clauses of CONNECT, ATTACH, and SET SESSION AUTHORIZATION, passing a mixed case password would not succeed even when the OpenVMS /FLAGS=PwdMix flag was specified for the user in the system authorization file (UAF).

In the example below, the OpenVMS user USER100 has /FLAGS=PwdMix defined in the UAF file.

```
SQL> attach 'filename personnel user ''user100'' using ''TEST100''';
%SQL-F-ERRATTDEC, Error attaching to database personnel
-RDB-E-AUTH_FAIL, authentication failed for user USER100
SQL> attach 'filename personnel user ''user100'' using ''Test100''';
SQL>
```

Support for this OpenVMS Version 7.3−2 feature is now included in Oracle Rdb Release 7.2.1.2. Refer to OpenVMS Version 7.3−2 documentation for more information on the PwdMix flag for user accounts.

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

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

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

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

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

Chapter 12
Enhancements And Changes Provided in Oracle Rdb Release 7.2.1.0

12.1 Enhancements And Changes Provided in Oracle Rdb Release 7.2.1.0

12.1.1 New Implementation of the CONCAT (||) Operator

This release of Oracle Rdb introduces a new CONCAT implementation that is more efficient and uses less virtual memory. This new implementation includes the following changes:

- The CONCAT built−in function now expects more than two parameters. These parameters may be any data type except LIST OF BYTE VARYING. Non−character string types are implicitly CAST as CHARACTER VARYING prior to processing.
- A series of || operators and CONCAT functions is now automatically collected and converted to this new CONCAT list operator. This will affect query outline ids and some query outlines may need to be recreated after installing this version.
- When the dialect is ORACLE LEVEL1 or ORACLE LEVEL2, the new CONCAT operator fully supports the Oracle semantics for concatanation. Namely, any value expression that results in NULL is ignored. The previous implementation rewrote the source query adding CASE expressions to implement these semantics which added considerable CPU overhead on some queries.

12.1.2 New Columns in Information Table RDB$JOURNALS

Bug 5401232

Columns RDB$SEQUENCE_NUMBER and RDB$STATE have been added to the Information Table RDB$JOURNALS. These contain the current AIJ sequence number and the STATE (either "Current" or "Latent") for the AIJ file.

```
SQL> select RDB$SEQUENCE_NUMBER, RDB$STATE from RDB$JOURNALS;
 RDB$SEQUENCE_NUMBER   RDB$STATE
                  −1   Latent
                  −1   Latent
                   3   Current
3 rows selected
```

To "upgrade" an existing database, which already contains the Information Table RDB$JOURNALS, simply drop the table and re−run SQL$SAMPLE:INFO_TABLES.SQL. This will create a new and complete RDB$JOURNALS table (any errors concerning the existence of any other Information Tables can be safely ignored). Or, drop the Information Table and recreate the Information Table with the required columns (see SQL$SAMPLE:INFO_TABLES.SQL for the available columns).

After installing Rdb Release 7.2.1, any database which contains an RDB$JOURNALS Information Table will continue to function as before but the new columns will not be visible. Also, previous versions of INFO_TABLES.SQL will still function as before but, again, the new columns will not be visible.

New column definitions for RDB$JOURNALS:

```
Column Name          Date Type   Domain
−−−−−−−−−−−          −−−−−−−−−   −−−−−−
```

```
RDB$SEQUENCE_NUMBER  integer     RDB$COUNTER
RDB$STATE            char(31)    RDB$USAGE
```

12.1.3 Oracle Rdb Release 7.2.x.x New Features Document Added

A new document has been created which contains all of the New Features Chapters from all previous Rdb 7.2 Release Notes. This document will be included in saveset A of the Rdb kit. It is called RDB_NEWFEATURES_72xx and will be available in postscript, text and PDF format. This will provide customers with one document to reference to find out about all new features that have been added to the Rdb 7.2 releases.

12.1.4 Hot Standby Status Symbols From RMU /SHOW AFTER_JOURNAL /BACKUP_CONTEXT

Additional DCL symbols indicating the Hot Standby replication state are now created by the RMU /SHOW AFTER_JOURNAL /BACKUP_CONTEXT command.

The symbol names are listed below:

- RDM$HOT_STANDBY_STATE − Contains the current replication state. Possible state strings and the description of each state are listed below:
  - "Inactive" − Inactive
  - "DB_Bind" − Binding to database
  - "Net_Bind" − Binding to network
  - "Restart" − Replication restart activity
  - "Connecting" − Waiting for LCS to connect
  - "DB_Synch" − Database synchronization
  - "Activating" − LSS server activation
  - "SyncCmpltn" − LRS synchronization redo completion
  - "Active" − Database replication
  - "Completion" − Replication completion
  - "Shutdown" − Replication cleanup
  - "Net_Unbind" − Unbinding from network
  - "Recovery" − Unbinding from database
  - "Unknown" − Unknown state or unable to determine state
- RDM$HOT_STANDBY_SYNC_MODE − Contains the current replication synchronization mode when replication is active. Possible synchronization mode strings are listed below:
  - "Cold"
  - "Warm"
  - "Hot"
  - "Commit"
  - "Unknown"

12.1.5 RMU BACKUP, COPY, MOVE /THREADS=n New Qualifier

A new qualifier has been added to allow the user to better control the system load created by a backup, copy or move operation. The new qualifier allows the user to specify the number of threads to be used by RMU.

RMU creates so called internal 'threads' of execution to read data from one specific storage area. Threads run quasi−parallel within the process executing the RMU image. Each thread generates its own I/O load and consumes resources like virtual address space and process quotas (e.g. FILLM, BYTLM). The more threads, the more I/Os can be generated at one point in time and the more resources are needed to accomplish the same task.

Performance increases with more threads due to parallel activities which keep disk drives busier. However, at a certain number of threads, performance suffers because the disk I/O subsystem is saturated and I/O queues build up for the disk drives. Also the extra CPU time for additional thread scheduling overhead reduces the overall performance. Typically 2−5 threads per input disk drive are sufficient to drive the disk I/O subsystem at its optimum. However, some controllers may be able to handle the I/O load of more threads, e.g. disk controllers with RAID sets and extra cache memory.

In a COPY or MOVE operation, one thread moves the data of one storage area at a time. If there are more storage areas to be moved than there are threads, then the next idle thread takes on the next storage area. Storage areas are moved in order of the area size, largest areas first. This optimizes the overall elapsed time by allowing other threads to move smaller areas while an earlier thread is still working on a large area. If no threads qualifier is specified, then 10 threads are created by default. The minimum is 1 thread and the maximum is the number of storage areas to be copied or moved. If the user specifies a value larger than the number of storage areas, then RMU silently limits the number of threads to the number of storage areas.

In a BACKUP operation, one writer thread is created per output stream. An output stream can be either a tape drive, a disk file or a media library manager stream. In addition, RMU creates a number of reader threads and their number can be specified. RMU assigns a subset of reader threads to writer threads. RMU calculates the assignment so that roughly the same amount of data is assigned to each output stream. By default, five reader threads are created for each writer thread. If the user has specified the number of threads, then this number is used to create the reader thread pool. RMU always limits the number of reader threads to the number of storage areas. A threads number of 0 causes RMU to create one thread per storage area which start to run all in parallel immediately. Even though this may sound like a good idea to improve performance, this approach causes performance to suffer for databases with a larger number (>10) of storage areas. For a very large number of storage areas (>800), this fails due to hard limitations in system resources like virtual address space.

For a COPY or MOVE operation, you can specify a threads number as low as 1. Using a threads number of 1 generates the smallest system load in terms of working set usage and disk I/O load. Disk I/O subsystems most likely can handle higher I/O loads. Using a slightly larger value than 1 typically results in faster execution time.

For a BACKUP operation, the smallest threads number you can specify is the number of output streams. This gurantees that each writer thread has at least one reader thread assigned to it and does not produce an empty save set. Using a threads number equal to the number of output streams generates the smallest system load in terms of working set usage and disk I/O load. Disk I/O subsystems most likely can handle higher I/O loads. Using a slightly larger value than the number of output streams (assigning more reader threads to a writer thread), typically results in faster execution time.

The old READER_THREAD_RATIO qualifier has been deprecated but is still accepted and works exactly the same as in previous versions.

Examples using the /THREADS qualifier:

```
Copying one storage area at a time:

$ RMU /COPY /THREADS=1 /LOG FOO BCK
%RMU-I-MOVTXT_04, Starting move of storage area ...
%RMU-I-MOVTXT_01, Completed move of storage area ...
%RMU-I-MOVTXT_05, Moved snapshot area file ...
%RMU-I-MOVTXT_04, Starting move of storage area ...
%RMU-I-MOVTXT_01, Completed move of storage area ...
%RMU-I-MOVTXT_05, Moved snapshot area file ...
    .
    .
    .
Copying three storage areas in parallel:

$ RMU /COPY /THREADS=3 /LOG FOO BCK
%RMU-I-MOVTXT_04, Starting move of storage area ...
%RMU-I-MOVTXT_04, Starting move of storage area ...
%RMU-I-MOVTXT_04, Starting move of storage area ...
%RMU-I-MOVTXT_01, Completed move of storage area ...
%RMU-I-MOVTXT_05, Moved snapshot area file ...
%RMU-I-MOVTXT_04, Starting move of storage area ...
%RMU-I-MOVTXT_01, Completed move of storage area ...
%RMU-I-MOVTXT_05, Moved snapshot area file ...
    .
    .
    .
```

12.1.6 Concealed Logical Names Defined in LNM$SYSCLUSTER_TABLE Table Allowed

Previously, many uses of concealed logical device names were required to be defined in the LNM$SYSTEM_TABLE logical name table. This requirement is in place to ensure that various components of the database system running in separate process contexts would all have access to the same logical name definitions. Uses of concealed logical device names that were not defined in the LNM$SYSTEM_TABLE could result in a *COSI-F-NOTSYSCONCEAL "non-system concealed device name in filename" status*.

This restriction has been somewhat relaxed. While all processes using a database still require access to the same logical name definitions, this can now be accomplished by using the LNM$SYSTEM_TABLE logical name table or the LNM$SYSCLUSTER_TABLE logical name table (which represents a cluster-wide resource). Note, however, that it is strongly recommended that concealed logical device names not be defined in both tables at the same time on any cluster node as this can lead to unpredictable results possibly leading to database corruption or instability.

12.1.7 Support for GNAT Ada on Alpha and Itanium

Support has been added to Precompiled SQL and SQL Module Language for the Ada Core GNAT Ada compiler. This support allows SQL$PRE/ADA compilations to target the GNAT Ada compiler and facilitates

interfacing SQL Module Language modules to GNAT Ada programs. For migrating existing applications from DEC Ada to GNAT Ada, in most cases the only changes needed are those required by the different rules of the two language variants. The most significant changes for most DEC Ada applications will be because GNAT Ada requires a source file to contain a single "compilation unit" which means a single package specification or a single package body. Files containing package specifications and bodies must use the suffixes .ADS and .ADB, respectively.

GNAT Ada uses a more Unix−like "compilation environment" in contrast to the Ada Development Library approach of DEC Ada. It consists of the following three steps: GNAT COMPILE which produces object files and .ALI files (Ada Library Information); GNAT BIND which checks consistency, determines the order of elaboration, and generates a main program which incorporates that elaboration; and GNAT LINK which compiles the main program from GNAT BIND, builds a set of linker options, and calls the OpenVMS link utility to produce an executable program. There is also a utility called GNAT MAKE which folds these steps together, including detecting obsolete programs and recompiling them. In most cases, Precompiled SQL applications and applications which call SQL Module Language modules can be built using GNAT MAKE provided that the .SQLADA and .SQLMOD source code files are compiled with SQL$PRE or SQL$MOD beforehand.

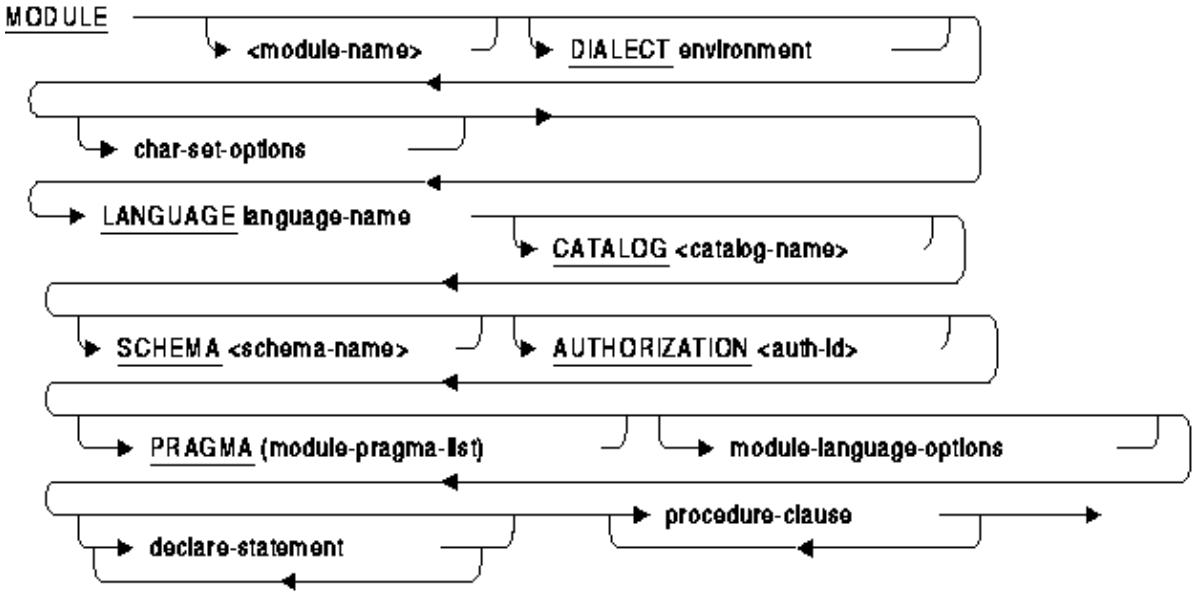
For information about GNAT Ada development see the Ada Core documentation for GNAT Ada on OpenVMS. The following specific Ada Core documents are pertinent:

- GNAT Pro User's Guide − OpenVMS − GNAT Pro Ada 95 Compiler
- GNAT Pro Reference Manual − GNAT Pro Ada 95 Compiler

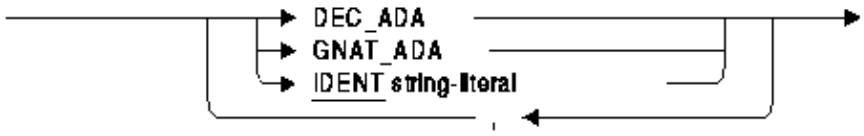
The minimum supported version of the Ada Core GNAT Ada compilers are as follows: for Alpha, 5.04a1 and for Itanium, 5.04a.

On Itanium, GNAT Ada is the only option for Ada development since DEC Ada is not supported by HP on Itanium. On Alpha, either DEC Ada or GNAT Ada may be used. For SQL$PRE, this choice is determined by values added to the /ADA qualifier as follows: /ADA={DEC_ADA,GNAT_ADA} where DEC_ADA is the default.

For SQL Module Language on Alpha, there are two means of specifying which Ada compiler is the target. First, there is a new clause in the module header called "PRAGMA" which can have valid keywords of GNAT_ADA and DEC_ADA (but only one of them). In the future, additional keywords may be added to the PRAGMA clause for other purposes. The pragma clause appears in the module header according to the following syntax:



module-pragma-list =



The following example shows the use of a PRAGMA clause in a module header to specify that the target is GNAT Ada:

```
MODULE          MY_MODULE
DIALECT         SQL99
LANGUAGE        ADA
AUTHORIZATION   SAMPLE_USER
PRAGMA          (GNAT_ADA)
ALIAS           RDB$DBHANDLE
PARAMETER       COLONS
```

The second method of specifying the target compiler is a new SQL$MOD qualifier /PRAGMA={GNAT_ADA,DEC_ADA} on the command line. A PRAGMA clause in the code takes precedence over the qualifier.

12.1.7.1 Pragma EXTEND_SYSTEM

SQL depends on certain types which are defined in package SYSTEM in DEC Ada. Many of these types are not in the Ada Core implementation. A special pragma exists in the Ada Core implementation which allows use of these types and their associated functions. It is as follows:

```
pragma EXTEND_SYSTEM (AUX_DEC);
```

This pragma can be added to the GNAT Ada compilation configuration file (GNAT.ADC) in your compilation directory and will automatically be applied to all compilations. See the Ada Core documentation for more information about the GNAT Ada compilation environment and the use of the GNAT.ADC file.

12.1.7.2 SQL_STANDARD Package

For DEC Ada, the package SQL_STANDARD is stored in a file named SQL$STANDARD.ADA which is placed in SYS$LIBRARY by the Rdb installation procedure. In order to conform to GNAT Ada naming conventions, a new file has been created to contain the SQL_STANDARD package. This file is SQL_STANDARD.ADS and is placed in SYS$LIBRARY by the Rdb install procedures.

12.1.7.3 GNAT Ada Type Differences

With GNAT Ada, the default address size for the SYSTEM.ADDRESS type is 64 bits. All SQL routines currently use 32 bit addresses. Accordingly, the definition of the Ada SQLVAR_REC record (which is used in the SQLDA) has been changed so the SQLDATA and SQLIND components are SYSTEM.SHORT_ADDRESS in lieu of SYSTEM.ADDRESS.

Other relevant GNAT Ada type differences have to do with floating point datatypes (corresponding to the SQL REAL and DOUBLE PRECISION datatypes). Instead of providing types for explicit floating point representation in package SYSTEM, GNAT Ada provides pragmas to specify the floating point representation for the types in package STANDARD. These pragmas are Float_Representation and Long_Float. Float_Representation allows you to specify IEEE_Float or VAX_Float. If you specify VAX_Float, pragma Long_Float allows you to specify D_Float or G_Float. The package STANDARD types which are relevant to SQL and affected be these pragmas are SINGLE_FLOAT and DOUBLE_FLOAT. See the Ada Core documentation for more information about the GNAT Ada floating point pragmas.

12.1.7.4 SQL Module Language

SQL$MOD generates and calls GNAT COMPILE to compile an Ada package specification which has the same name as the module and the suffix ".ADS". For the example module header above, the package specification file would be "MY_MODULE.ADS". GNAT COMPILE creates an object file which, continuing the example, would be named "MY_MODULE.OBJ". The SQL object file generated directly by SQL$MOD will, by default, have the same prefix as the .SQLMOD source file and a suffix of .OBJ or, if the "/OBJECT=" qualifier is used, the name specified by that qualifier. SQL$MOD will detect if this name duplicates the name of the object file out of the GNAT Ada compiler and, if so, will use a .SQL_OBJ suffix for its object file to avoid the name conflict. SQL$MOD also generates a VMS linker utility options file which allows GNAT LINK to link in the SQL object file to the application. This options file, named "module_name.OPT", also contains an entry for the SQL$USER library so that GNAT LINK will be able to resolve the references to it in the SQL$MOD−generated object file. SQL$MOD adds a "pragma Linker_Options" to the generated .ADS file to tell GNAT LINK about the generated .OPT file. GNAT LINK integrates the SQL$MOD−generated options file into the options file that it creates for the OpenVMS linker utility. This approach allows most GNAT Ada applications which call SQL Module Language modules to be built using the GNAT MAKE utility once the SQL$MOD compile is completed.

12.1.7.5 Precompiled SQL

Precompiled SQL generates several output files for a .SQLADA source file. One of these is the Ada source file which contains the Ada code in the original SQLADA file with the EXEC SQL statements translated into procedure calls. With DEC Ada, this file has the same name as the original SQLADA file but with a .ADA extension. When targeting GNAT Ada, this file has an extension of .ADB in order to conform to GNAT Ada naming conventions. SQL$PRE calls the GNAT compiler to compile the .ADB file into an object file with the .OBJ suffix and the .ALI file needed by the GNAT BIND and GNAT LINK commands. As with DEC Ada, SQL$PRE replaces the EXEC SQL statements with calls to routines in a SQL module. SQL$PRE produces an object file for the SQL module and an Ada package specification for it. As with DEC Ada, the default file name prefix for generated SQL module files is formed by prefixing "SQL_" on the original file name. The only difference is that the Ada package spec for the SQL module has the suffix ".ADS" in conformance with GNAT Ada conventions.

When targeting GNAT Ada, SQL$PRE automatically calls the GNAT Ada compiler with the generated Ada files just as it does for DEC Ada. Both the .ADB and .ADS files generated by SQL$PRE are compiled so that an executable can be built using the GNAT BIND and GNAT LINK commands. The compilation of the .ADS file results in an object file which would have the same name as the object file generated by SQL$PRE, that is: "SQL_module_name.OBJ". Accordingly, the object file generated by SQL$PRE is named: "SQL_module_name.SQL_OBJ". SQL$PRE generates a VMS linker utility options file which allows GNAT LINK to link in the .SQL_OBJ file to the application. This options file, named "SQL_module_name.OPT", also contains an entry for the SQL$USER library so that GNAT LINK will be able to resolve the references to it in the SQL$PRE−generated object file. A "pragma Linker_Options" is added to the .ADS file to tell GNAT LINK about the generated .OPT file.

For GNAT Ada, the declaration of RDB_MESSAGE_VECTOR as an object mapping to the RDB$MESSAGE_VECTOR PSECT has been moved to the .ADB file because the generated code will not link properly if this declaration is in a .ADS file.

The following example shows building a Precompiled SQL application using the GNAT Ada compiler. Note that on Itanium, the "=/GNAT_ADA" qualifier would be unnecessary because only GNAT Ada is supported on that platform. This example shows how to build and run the Ada version of the SQL_ALL_DATATYPES application from SQL$SAMPLES.

```
$!
$! Set up GNAT Ada environment
$!
$ CREATE GNAT.ADC
pragma EXTEND_SYSTEM (AUX_DEC);
$ DEFINE ADA_INCLUDE_PATH SYS$LIBRARY
$ GNAT COMPILE SYS$LIBRARY:SQL_STANDARD.ADS
$!
$! Build SQL_ALL_DATATYPES
$!
$ SQL$PRE/ADA=GNAT_ADA SQL$SAMPLE:SQL_ALL_DATATYPES
$ GNAT MAKE SQL_ALL_DATATYPES
$!
$ RUN SQL_ALL_DATATYPES
```

12.1.8 Enhancement to SQLCA

The following enhancements have been made to the SQLCA with this release:

- The SQLCA field SQLERRD[0] is now updated with the statement type by the PREPARE statement for all dialects. These numeric codes are listed in the table below.
  In previous releases, SQLERRD[0] was set only for ORACLE LEVEL1 and ORACLE LEVEL2 dialects.
- If the statement being prepared is a SELECT statement containing an INTO clause, then SQLCA field SQLWARN6 will contain the character "I". Such singleton SELECT statements can be executed without using a cursor.

Table 12−1 SQLCA SQLERRD [0] Values

Symbolic Name+	Value	SQL Statement
	0	Statement is unknown
SQL_K_OCTRDB_CONNECT	−1	Rdb Connect
SQL_K_OCTRDB_ATTACH	−2	Rdb Attach
SQL_K_OCTRDB_DISCONNECT	−3	Rdb Disconnect
SQL_K_OCTRDB_CREATE_MODULE	−4	Rdb Create Module
SQL_K_OCTRDB_ALTER_MODULE	−5	Rdb Alter Module
SQL_K_OCTRDB_DROP_MODULE	−6	Rdb Drop Module
SQL_K_OCTRDB_CREATE_DOMAIN	−7	Rdb Create Domain
SQL_K_OCTRDB_ALTER_DOMAIN	−8	Rdb Alter Domain
SQL_K_OCTRDB_DROP_DOMAIN	−9	Rdb Drop Domain
SQL_K_OCTRDB_CREATE_CATALOG	−10	Rdb Create Catalog
SQL_K_OCTRDB_ALTER_CATALOG	−11	Rdb Alter Catalog
SQL_K_OCTRDB_DROP_CATALOG	−12	Rdb Drop Catalog
SQL_K_OCTRDB_ALTER_SCHEMA	−13	Rdb Alter Schema
SQL_K_OCTRDB_DROP_SCHEMA	−14	Rdb Drop Schema
SQL_K_OCTRDB_SET_SESSION	−15	Rdb Set Session Authorization
SQL_K_OCTCTB	1	create table
SQL_K_OCTINS	2	insert
SQL_K_OCTSEL	3	select
SQL_K_OCTCCL	4	create cluster
SQL_K_OCTACL	5	alter cluster
SQL_K_OCTUPD	6	update
SQL_K_OCTDEL	7	delete
SQL_K_OCTDCL	8	drop cluster
SQL_K_OCTCIX	9	create index
SQL_K_OCTDIX	10	drop index
SQL_K_OCTAIX	11	alter index
SQL_K_OCTDTB	12	drop table

SQL_K_OCTCSQ	13	create sequence
SQL_K_OCTASQ	14	alter sequence
SQL_K_OCTATB	15	alter table
SQL_K_OCTDSQ	16	drop sequence
SQL_K_OCTGRA	17	grant
SQL_K_OCTREV	18	revoke
SQL_K_OCTCSY	19	create synonym
SQL_K_OCTDSY	20	drop synonym
SQL_K_OCTCVW	21	create view
SQL_K_OCTDVW	22	drop view
SQL_K_OCTVIX	23	validate index
SQL_K_OCTCPR	24	create procedure
SQL_K_OCTAPR	25	alter procedure
SQL_K_OCTLTB	26	lock table
SQL_K_OCTNOP	27	no operation
SQL_K_OCTRNM	28	rename
SQL_K_OCTCMT	29	comment
SQL_K_OCTAUD	30	audit
SQL_K_OCTNOA	31	noaudit
SQL_K_OCTCED	32	create database link
SQL_K_OCTDED	33	drop database link
SQL_K_OCTCDB	34	create database
SQL_K_OCTADB	35	alter database
SQL_K_OCTCRS	36	create rollback segment
SQL_K_OCTARS	37	alter rollback segment
SQL_K_OCTDRS	38	drop rollback segment
SQL_K_OCTCTS	39	create tablespace
SQL_K_OCTATS	40	alter tablespace
SQL_K_OCTDTS	41	drop tablespace
SQL_K_OCTASE	42	alter session
SQL_K_OCTAUR	43	alter user
SQL_K_OCTCWK	44	commit
SQL_K_OCTROL	45	rollback
SQL_K_OCTSPT	46	savepoint
SQL_K_OCTPLS	47	pl/sql execute
SQL_K_OCTSET	48	set transaction
SQL_K_OCTASY	49	alter system switch log
SQL_K_OCTXPL	50	explain
SQL_K_OCTCUS	51	create user
SQL_K_OCTCRO	52	create role
SQL_K_OCTDUS	53	drop user
SQL_K_OCTDRO	54	drop role

SQL_K_OCTSER	55	set role
SQL_K_OCTCSC	56	create schema
SQL_K_OCTCCF	57	create control file
SQL_K_OCTATR	58	Alter tracing
SQL_K_OCTCTG	59	create trigger
SQL_K_OCTATG	60	alter trigger
SQL_K_OCTDTG	61	drop trigger
SQL_K_OCTANT	62	analyze table
SQL_K_OCTANI	63	analyze index
SQL_K_OCTANC	64	analyze cluster
SQL_K_OCTCPF	65	create profile
SQL_K_OCTDPF	66	drop profile
SQL_K_OCTAPF	67	alter profile
SQL_K_OCTDPR	68	drop procedure
SQL_K_OCTARC	70	alter resource cost
SQL_K_OCTCSL	71	create snapshot log
SQL_K_OCTASL	72	alter snapshot log
SQL_K_OCTDSL	73	drop snapshot log
SQL_K_OCTCSN	74	create snapshot
SQL_K_OCTASN	75	alter snapshot
SQL_K_OCTDSN	76	drop snapshot
SQL_K_OCTCTY	77	create type
SQL_K_OCTDTY	78	drop type
SQL_K_OCTARO	79	alter role
SQL_K_OCTATY	80	alter type
SQL_K_OCTCYB	81	create type body
SQL_K_OCTAYB	82	alter type body
SQL_K_OCTDYB	83	drop type body
SQL_K_OCTDLB	84	drop library
SQL_K_OCTTTB	85	truncate table
SQL_K_OCTTCL	86	truncate cluster
SQL_K_OCTCBM	87	create bitmapfile
SQL_K_OCTAVW	88	alter view
SQL_K_OCTDBM	89	drop bitmapfile
SQL_K_OCTSCO	90	set constraints
SQL_K_OCTCFN	91	create function
SQL_K_OCTAFN	92	alter function
SQL_K_OCTDFN	93	drop function
SQL_K_OCTCPK	94	create package
SQL_K_OCTAPK	95	alter package
SQL_K_OCTDPK	96	drop package
SQL_K_OCTCPB	97	create package body

SQL_K_OCTAPB	98	alter package body
SQL_K_OCTDPB	99	drop package body
SQL_K_OCTCDR	157	create directory
SQL_K_OCTDDR	158	drop directory
SQL_K_OCTCLB	159	create library
SQL_K_OCTCJV	160	create java
SQL_K_OCTAJV	161	alter java
SQL_K_OCTDJV	162	drop java
SQL_K_OCTCOP	163	create operator
SQL_K_OCTCIT	164	create indextype
SQL_K_OCTDIT	165	drop indextype
SQL_K_OCTAIT	166	reserver for alter indextype
SQL_K_OCTDOP	167	drop operator
SQL_K_OCTAST	168	associate statistics
SQL_K_OCTDST	169	disassociate statistics
SQL_K_OCTCAL	170	call method
SQL_K_OCTCSM	171	create summary
SQL_K_OCTASM	172	alter summary
SQL_K_OCTDSM	173	drop summary
SQL_K_OCTCDM	174	create dimension
SQL_K_OCTADM	175	alter dimension
SQL_K_OCTDDM	176	drop dimension
SQL_K_OCTCCT	177	create context
SQL_K_OCTDCT	178	drop context
SQL_K_OCTASO	179	alter outline
SQL_K_OCTCSO	180	create outline
SQL_K_OCTDSO	181	drop outline
SQL_K_OCTAOP	183	alter operator
SQL_K_OCTCEP	184	create encryption profile
SQL_K_OCTAEP	185	alter encryption profile
SQL_K_OCTDEP	186	drop encryption profile
SQL_K_OCTCSP	187	create spfile from pfile
SQL_K_OCTCPS	188	create pfile from spfile
SQL_K_OCTUPS	189	merge
SQL_K_OCTCPW	190	change password
SQL_K_OCTUJI	191	update join index
SQL_K_OCTASYN	192	alter synonym
SQL_K_OCTADG	193	alter disk group
SQL_K_OCTCDG	194	create disk group
SQL_K_OCTDDG	195	drop disk group
SQL_K_OCTALB	196	alter library
SQL_K_OCTPRB	197	purge user recyclebin

SQL_K_OCTPDB	198	purge dba recyclebin
SQL_K_OCTPTS	199	purge tablespace
SQL_K_OCTPTB	200	purge table
SQL_K_OCTPIX	201	purge index
SQL_K_OCTUDP	202	undrop object
SQL_K_OCTDDB	203	drop database
SQL_K_OCTFBD	204	flashback database
SQL_K_OCTFBT	205	flashback table

+ The positive values are defined for compatibility with Oracle 10g. Not all statements are supported by Oracle Rdb therefore not all values will appear in the SQLCA. Negative values are Oracle Rdb specific values.

12.1.9 File−System Caching Avoided for RMU /COPY, /MOVE, /BACKUP And /RESTORE, IO To Database

In order to reduce CPU consumption and XFC spinlock contention and to help avoid "thrashing" the file system cache and to streamline file read and write operations, caching by the operating system is disabled for various files and operations including:

- RMU /COPY
- RMU /MOVE
- RMU /BACKUP
- RMU /RESTORE
- Most Database Root File IO
- Most Database RUJ File IO
- Most Row−Cache Backing Store File IO
- Most Recovery Work File IO

Testing on various configurations indicates that, in general, avoiding the operating system's XFC cache for these database file IO operations results in better overall performance as balanced between CPU and IO costs.

12.1.10 RMU /BACKUP /COMPRESSION New Algorithm

The RMU /BACKUP /COMPRESSION feature has been enhanced to offer an additional compression algorithm. The ZLIB algorithm and software, developed by Jean−loup Gailly and Mark Adler, has been implemented for RMU /BACKUP /COMPRESS. This implementation generally uses the same or less CPU time and is generally more effective (compresses better) than either of the HUFFMAN or LZSS algorithms.

The /COMPRESSION qualifier accepts the following keywords:

- HUFFMAN − HUFFMAN encoding algorithm.
- LZSS − Lempel−Ziv algorithm.
- ZLIB=level − ZLIB algorithm. The "level" value is an integer between 1 and 9 specifying the relative compression level with one being the least amount of compression and nine being the greatest amount of compression. Higher levels of the compression use increased CPU time while generally providing

better compression. The default compression level of 6 is a balance between compression effectiveness and CPU consumption.

If you specify the /COMPRESSION qualifier without a value, the default is /COMPRESSION=ZLIB=6.

Here are examples using the /COMPRESS qualifier. Note that if "/LOG=FULL" is specified, data compression statistics information is displayed.

```
$ RMU /BACKUP /COMPRESS /NOLOG FOO BCK
$ RMU /BACKUP /COMPRESS=ZLIB:9 /LOG=FULL FOO BCK
    .
    .
    .
BACKUP summary statistics:
        Data compressed by 53% (9791 KB in/4650 KB out)
```

Older Oracle Rdb 7.2 Releases and Compressed RBF Files

Prior releases of Oracle Rdb are unable to read RBF files compressed with the ZLIB algorithm. In order to read compressed backups with Oracle Rdb Release 7.2 prior to V7.2.1, they must be made with /COMPRESSION=LZSS or /COMPRESSION=HUFFMAN explicitly specified (because the default compression algorithm has been changed from LZSS to ZLIB). Oracle Rdb Release 7.2.1 is able to read compressed backups using the LZSS or HUFFMAN algorithms made with prior releases.

Compression Effectiveness Varies

The actual amount of compression for any algorithm is strongly dependent on the actual data being compressed. Some database content may compress quite well and other content may compress not at all and may actually result in expansion of the output.

When using the /ENCRYPT and /COMPRESS features together, data is first compressed and then encrypted. This provides effective compression as well as effective encryption.

12.1.11 Enhancements for Compression Support in RMU Unload and Load

Bugs 690179 and 675012

This release of Oracle Rdb introduces support for compression to RMU Unload, RMU Load and RMU Dump Export.

Data compression is applied to the user data unloaded to the internal (interchange) format file. Table rows, null byte vector and LIST OF BYTE VARYING data is compressed using either the LZW (Lempel−Ziv−Welch) technique or the ZLIB algorithm developed by Jean−loup Gailly and Mark Adler. Table metadata (column names and attributes) are never compressed and the resulting file remains a structured interchange file. This file can also be processed using the RMU Dump Export command.

In past releases, it was possible that table data, stored in the database with compression enabled, would be

many times smaller in the database than when unloaded by RMU. In the database, a simple and fast RLE (run−length encoding) algorithm is used to store rows but this data is fully expanded by RMU Unload. Allowing compression allows the result data file to be more compact using less disk space and permitting faster transmission over communication lines.

Changes to RMU Unload

A new /COMPRESSION qualifier has been added to RMU Unload. The default remains /NOCOMPRESSION. This qualifier accepts the following optional keywords: LZW, ZLIB, LEVEL and EXCLUDE_LIST. The compression algorithms used are ZLIB (the default) or LZW. ZLIB allows further tuning with the LEVEL option that accepts a numeric level between 1 and 9. The default of 6 is usually a good trade off between result file size and the CPU cost of the compression.

It is possible that data in LIST OF BYTE VARYING columns is already in a compressed format (for instance images as JPG data) and therefore need not be compressed by RMU Unload. In fact, compression in such cases might actually cause the output to grow. Therefore, the /COMPRESSION qualifier accepts an option EXCLUDE_LIST which will disable compression for LIST OF BYTE VARYING columns. Specific column names can be listed or, if omitted, all LIST OF BYTE VARYING columns will be excluded from compression.

```
$ rmu/unload/compress=LZW/debug=trace complete_works complete_works
complete_works
Debug = TRACE
Compression = LZW
* Synonyms are not enabled
Unloading Blob columns.
Row_Count = 500
Message buffer: Len: 54524
Message buffer: Sze: 109, Cnt: 500, Use: 31 Flg: 00000000
** compress data: input 2700 output 981 deflate 64%
** compress TEXT_VERSION : input 4454499 output 1892097 deflate 58%
** compress PDF_VERSION : input 274975 output 317560 deflate −15%
%RMU−I−DATRECUNL,   30 data records unloaded.
```

In this example, the column PDF_VERSION contains data that does not compress and so should be excluded on the command line.

```
$ rmu/unload/compress=(LZW,exclude_list:PDF_VERSION)/debug=trace complete_works
complete_works complete_works
Debug = TRACE
Compression = LZW
Exclude_List:
        Exclude column PDF_VERSION
* Synonyms are not enabled
Unloading Blob columns.
Row_Count = 500
Message buffer: Len: 54524
Message buffer: Sze: 109, Cnt: 500, Use: 31 Flg: 00000000
** compress data: input 2700 output 981 deflate 64%
** compress TEXT_VERSION : input 4454499 output 1892097 deflate 58%
%RMU−I−DATRECUNL,   30 data records unloaded.
```

Note

Short rows and short null byte vectors will cause compression to be automatically disabled

for the table. However, compression of LIST OF BYTE VARYING columns will still be performed.

The /COMPRESSION qualifier accepts either LZW or ZLIB as the compression method. ZLIB is the default and tends to do the best general compression. However, after testing, the database administrator may decide to use LZW method.

Changes to RMU Load

No new qualifiers are required by RMU Load. The metadata in the interchange file defines the compression algorithm used by RMU Load and indicates which LIST OF BYTE VARYING columns were compressed by RMU Unload.

Changes to RMU Dump Export

A new /OPTIONS qualifier has been added to RMU Dump Export. The default is NOOPTIONS. It accepts the keyword HEADER_SECTION which allows the database administrator to display just the header portion of the interchange file and avoid dumping the data or metadata for every row in the table.

```
$ RMU/DUMP/EXPORT/OPTION=HEADER JOBS.UNL

BEGIN HEADER SECTION - (0)
    NONCORE_TEXT HDR_BRP_ID - (20) :  Load/Unload utility
    CORE_NUMERIC HDR_BRPFILE_VERSION - (1) : 4
    NONCORE_TEXT HDR_DBS_ID - (18) : Oracle Rdb V7.2-10
    NONCORE_TEXT HDR_DB_NAME - (16) : DB$:MF_PERSONNEL
    NONCORE_DATE HDR_DB_LOG_BACKUP_DATE - (8) :  3-JUL-2006 16:52:32.83
    CORE_NUMERIC HDR_DATA_COMPRESSION - (1) : 1
END HEADER SECTION - (0)
```

Here HDR_DATA_COMPRESSION indicates that compression has been used by RMU Unload.

Usage Notes

- Only the user data is compressed. Therefore, additional compression may be applied using various third party compression tools, such as ZIP. It is not the goal of RMU to replace such tools.
- The qualifier RECORD_DEFINITION (or RMS_RECORD_DEF) is not compatible with /COMPRESSION. Note that the TRIM option for DELIMITED format can be used to trim trailing spaces from VARCHAR data.
- The LEVEL keyword may not be used with LZW compression technique. Only one of LZW or ZLIB may be specified for the /COMPRESSION qualifier.

12.1.12 RMU /UNLOAD /AFTER_JOURNAL Commit Information Includes Username

Enhancement 5128647

The Oracle Rdb LogMiner (tm) feature is now able to extract username information. The "C"ommit information record has been extended to include a 12 byte username string. To specify that commit information records are to be extracted to the output stream, specify the COMMIT keyword to the

/INCLUDE=ACTION=qualifier.

In DUMP output format, a new field RDB$LM_USERNAME includes the username information. In DELIMITED_TEXT format, a new field is included at the end of the existing record. In TEXT and BINARY format, the username field is added as 12 bytes at the end of the record.

The following example demonstrates using the DUMP output format.

```
$ RMU /UNLOAD /AFTER_JOURNAL SQL$DATABASE AIJ1.AIJBCK -
        /INCLUDE=ACTION=COMMIT -
        /FORMAT=DUMP -
        /TABLE=(NAME=FOO,OUTPUT=FOO.DAT)
$ SEARCH /NOHEAD FOO.DAT RDB$LM_USERNAME
RDB$LM_USERNAME                 : JONES
RDB$LM_USERNAME                 : SMYTHE
```

12.1.13 SHOW DOMAIN and SHOW TABLE Have Better Formatting of DEFAULT Strings

The output from the SHOW DOMAIN and SHOW TABLE command has changed with respect to the DEFAULT values that are strings. In prior versions, the default string was displayed without delimiters which made it hard to read, especially if the default value was all spaces. Additionally, strings from different character sets were not identified.

This release of SQL now displays these strings in quotes and prefixes it with the character set name, unless the character set is the session default.

The following example shows the revised output.

```
SQL> show domain STREET_NAME;
STREET_NAME                     CHAR(40)
 Oracle Rdb default: '>>'
SQL>
SQL> show table (column) PERSON;
Information for table PERSON

Columns for table PERSON:
Column Name                     Data Type        Domain
-----------                     ---------        ------
LAST_NAME                       CHAR(50)
 Oracle Rdb default: ' '
LATIN_NAME                      VARCHAR(30)
         ISOLATIN1 30 Characters,  30 Octets
 Oracle Rdb default: ISOLATIN1' '

SQL>
```

12.1.14 CALL Statement From Trigger Action Can Now Update Tables

Bug 2421356

In prior releases of Oracle Rdb, the CALL statement could only SELECT data from other tables. With this release of Rdb, the CALL statement may INSERT, DELETE and UPDATE tables as well as CALL other routines. The following restrictions apply to the actions of the routines activated by the CALL statement:

- The table which is the target for the trigger, known as the morphing table, may not be updated (meaning INSERT, DELETE or UPDATE) by any stored procedure or function called within the scope of the trigger activation. Morphing table updates must be done within a trigger definition so that anomalies can be detected and avoided. Attempts to update the morphing tables will result in a runtime error such as the following:

```
%RDB-E-READ_ONLY_REL, relation T was reserved for read access; updates not
allowed
-RDMS-E-RTN_ERROR, routine "SET_LENGTH" generated an error during execution
-RDMS-F-INVTRGACT_STMT, invalid trigger action statement - can not modify
target table
```

  As far as the stored procedure is concerned, the morphing table is a read-only.
- If a stored routine action causes a different trigger to be activated and that then causes the same routine to be called, then an error similar to the following will be raised:

```
%RDB-F-ACTIVE_RTN, routine "CAST_VALUE" is already active
-RDMS-E-NORECURSION, no recursive routine calls permitted
```

Note

A stored routine may only be called from a trigger if it has been analyzed by Oracle Rdb. This step is automatically done by CREATE and ALTER TRIGGER ... ADD statements. If the routine was not recently created in the database (since Oracle Rdb Release 7.0.6), then use the ALTER MODULE ... COMPILE option to recompile any routines.

12.1.15 Using OpenVMS Reserved Memory Registry With Rdb

For Oracle Rdb memory-resident global sections (either row cache global sections or the database root global section), it is possible to utilize the OpenVMS Reserved Memory Registry feature to reserve physical memory. This reserved memory can be useful to allow the use of granularity hint (GH) regions which can further improve performance by using fewer processor translation buffer entries to map a large range of physical memory pages. Use of the reserved memory is optional and any performance gains are application specific.

In order to take advantage of the OpenVMS Reserved Memory Registry feature, global sections must be configured as "SHARED MEMORY IS PROCESS RESIDENT". This can be done with SQL statements "ALTER CACHE ... SHARED MEMORY IS PROCESS RESIDENT" and "ALTER DATABASE ... SHARED MEMORY IS PROCESS RESIDENT".

The name of the global section is required in order to register a global section in the OpenVMS shared memory registry. The "RMU/DUMP/HEADER" command can be used to display the global section names for the database root global section and the row cache global sections. This command also displays the size of the global sections in megabytes rounded up to the next whole megabyte.

For example, information about a row cache global section in the output from the RMU/DUMP/HEADER command might include the following:

```
Shared Memory...
  − Shared memory will be mapped resident
  − Global Section Name is "RDM72R$1$DGA2031064003D000000000005"
  − Shared memory section requirement is 77,070,336 bytes (74MB)
```

Information about the database global section in the output from the RMU/DUMP/HEADER command might include the following:

```
Derived Data...
  − Global section size
      With global buffers disabled is 2,047,042 bytes (2MB)
      With global buffers enabled is 33,860,114 bytes (33MB)
       .
       .
       .
  − Global Section Name is "RDM72N$1$DGA2031064003D000000000000"
```

From these examples, the row cache section size would be 74 megabytes and the database global section size (with global buffers enabled) would be 33 megabytes.

To reserve the memory, use the SYSMAN utility RESERVED_MEMORY ADD command and then run AUTOGEN as in the following examples:

```
$ RUN SYS$SYSTEM:SYSMAN
SYSMAN> RESERVED_MEMORY ADD RDM72N$1$DGA2031064003D000000000000 −
        /ALLOCATE /SIZE=33
SYSMAN> RESERVED_MEMORY ADD RDM72R$1$DGA2031064003D000000000005 −
        /ALLOCATE /SIZE=74
SYSMAN> EXIT
$ @SYS$UPDATE:AUTOGEN ...
```

The OpenVMS system must then be shutdown and restarted for the memory reservations to be in effect.

After rebooting and reopening databases, the SHOW MEMORY /RESERVED command can be used to see that the reserved memory is in use. For example:

```
$ SHOW MEMORY/RESERVED
Memory Reservations (pages):       Group  Reserved   In Use       Type
 RDM72R$1$DGA408451A6A000000000002
                                   SYSGBL         2        2  Page Table
 RDM72R$1$DGA408451A6A000000000002
                                   SYSGBL      1536     1353   Allocated
 Total (12.01 MBytes reserved)                 1538     1355
```

Database Root File Specific

Changes to the size of the database or row cache global sections will require that the memory reservation size be updated (either by removing and re−adding or modifying the existing reservation). Further, because the device and file identification of the database root file are encoded in the global section names, any operation (such as restoring or moving) that changes either the file identification or the device identification of the root file will result in the global section names changing.

If the reserved memory is specified with a size smaller than the actual size of the global section, the section may fail to be created when the database is opened or accessed with a message similar to "SYSTEM−F−INSFLPGS, insufficient Fluid Pages available".

For further information, review the OpenVMS documentation set including "HP OpenVMS System Manager's Manual, Volume 2: Tuning, Monitoring, and Complex Systems", "HP OpenVMS Version 8.2−1 for Integrity Servers New Features and Release Notes", and "HP OpenVMS System Services Reference Manual".

12.1.16 Server Output File Names As Database Attributes

Previously, logical names could be used to control various server output or log file names and locations. In many cases, these logical names would have to be defined system−wide and thus could effect the servers of multiple databases.

This situation has been improved. The output or log file names for a number of database server processes are now also controlled by optional database attributes.

The "RMU /SET SERVER /OUTPUT=filespec servertype" command can be used to specify the default output file specification for several of the database server processes. Existing logical names are still valid and supported and will override the database attribute if defined. If the output file specification is empty, the entry is disabled.

Note

The RMU /SET SERVER commands should only be used on the Master database. Do not do any RMU /SET SERVER commands on the Standby database as that updates the root and Hot Standby will no longer start.

Valid values for the "servertype" parameter and the matching logical name are:

Table 12−2 Server Types and Logical Names

Server	Servertype	Logical Name
AIJ Backup Server	ABS	RDM$BIND_BIND_ABS_OUTPUT_FILE
AIJ Log Server	ALS	RDM$BIND_BIND_ALS_OUTPUT_FILE
AIJ Log Roll−Forward Server	LRS	RDM$BIND_LRS_OUTPUT_FILE
AIJ Log Catch−Up Server	LCS	RDM$BIND_LCS_OUTPUT_FILE
Database Recovery Server	DBR	RDM$BIND_DBR_LOG_FILE
Row Cache Server	RCS	RDM$BIND_RCS_LOG_FILE

The /LOG qualifier can be used to display a log message at the completion of the RMU /SET operation.

Examples of using the "RMU /SET SERVER /OUTPUT=filespec servertype" command follow.

```
$ RMU /SET SERVER RCS /OUTPUT=RCS_PID.LOG /LOG DUA0:[DB]MYDB.RDB
$ RMU /SET SERVER ALS /OUTPUT=ALS$LOGS:ALS_DB1.LOG DUA0:[DB1]MFP.RDB
```

```
$ RMU /SET SERVER LRS /OUTPUT="" DUA0:[ZDB]ZDB.RDB
$ RMU /SET SERVER DBR /OUTPUT=DBR$LOGS:DBR.LOG DUA0:[ADB]ADB.RDB
```

12.1.17 New REBLDSPAM Informational Message Added to RMU/VERIFY

An informational "REBLDSPAM" message has been added to RMU/VERIFY which is output when the database Area Inventory (AIP) pages, which contain information about the database logical areas where table data and indexes are stored, are verified. This message is output if a flag is set for a logical area that indicates that the Space Management (SPAM) pages for the logical area should be updated to reflect changes to space thresholds or record lengths that may have been made by an SQL ALTER TABLE or other command. This message is INFORMATIONAL since updating the SPAM pages is not essential for the integrity or functionality of the database but can improve performance. The SPAM pages can be updated using the RMU/REPAIR command.

The following shows an example of this message which includes the name and id number of the logical area where the SPAM pages should be rebuilt to improve performance.

```
$RMU/VERIFY/ALL MF_PERSONNEL
%RMU-I-REBLDSPAM, Space management (SPAM) pages should be rebuilt for
                  logical area DEPARTMENTS_INDEX,
                  logical area id 74
%RMU-I-REBLDSPAM, Space management (SPAM) pages should be rebuilt for
                  logical area DEPARTMENTS,
                  logical area id 75
```

12.1.18 Increased Date/Time String Display Precision

For several values where there is enough space on the display, the RMU SHOW STATISTICS Utility now displays time/date stamps with precisions greater than 0.01 second units. In several cases (stall displays, for example), the screen display width must be 100 or more columns in order to display the full date/time with seven fractional digits.

For example, the "short" time and/or date format displays include only two fractional digits:

- 16:23:16.17
- 13-NOV-2006 16:23:16.17

While the "long" time and/or date format displays include seven fractional digits:

- 16:23:16.1776975
- 13-NOV-2006 16:23:16.1776975

12.1.19 Enhanced System Table Lookup in Multischema Databases

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

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

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

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

The same query in an older version would fail.

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

This problem has been corrected in Oracle Rdb Release 7.2.1.

12.1.20 New SET FLAGS Option: REBUILD_SPAM_PAGES

A new flag, REBUILD_SPAM_PAGES, has been added for use in conjunction with the DDL commands ALTER TABLE, ALTER STORAGE MAP, and ALTER INDEX.

When changing the row length or THRESHOLDS clause for a table or index, the corresponding SPAM pages for the logical area may require rebuilding. By default, these DDL commands update the AIP and set a flag to indicate that the SPAM pages should be rebuilt. However, this new flag may be set prior to executing a COMMIT for the transaction and the rebuild will take place within this transaction.

Use SET FLAGS 'NOREBUILD_SPAM_PAGES' to negate this flag.

The following example shows a simple change to the EMPLOYEES table (mapped in this example to a set of UNIFORM areas). The flag STOMAP_STATS is used to enable more trace information from the ALTER and COMMIT statements.

```
SQL> set transaction read write;
SQL>
SQL> set flags 'stomap_stats';
SQL>
SQL> alter table EMPLOYEES
cont>     add column MANAGERS_COMMENTS varchar(300);
~As: reads: async 0 synch 94, writes: async 18 synch 1
SQL>
SQL> alter storage map EMPLOYEES_MAP
cont>     store
cont>         using (EMPLOYEE_ID)
cont>             in EMPIDS_LOW
cont>           (thresholds (34,76,90))
cont>                 with limit of ('00200')
cont>             in EMPIDS_MID
cont>           (thresholds (34,76,90))
cont>                 with limit of ('00400')
cont>             otherwise in EMPIDS_OVER
cont>           (thresholds (34,76,90));
~As locking table "EMPLOYEES" (PR -> PU)
~As: removing superseded routine EMPLOYEES_MAP
~As: creating storage mapping routine EMPLOYEES_MAP (columns=1)
~As: reads: async 0 synch 117, writes: async 56 synch 0
SQL>
SQL> set flags 'rebuild_spam_pages';
SQL>
SQL> commit;
%RDMS-I-LOGMODVAL,     modified record length to 423
%RDMS-I-LOGMODVAL,     modified space management thresholds to (34%, 76%, 90%)
%RDMS-I-LOGMODVAL,     modified record length to 423
%RDMS-I-LOGMODVAL,     modified space management thresholds to (34%, 76%, 90%)
%RDMS-I-LOGMODVAL,     modified record length to 423
%RDMS-I-LOGMODVAL,     modified space management thresholds to (34%, 76%, 90%)
SQL>
```

The message LOGMODVAL will appear for each logical area in the storage map, one per partition.

This rebuild action only applies to UNIFORM storage areas and may incur significant I/O as SPAM pages and data pages are read to allow the SPAM page to be rebuilt.

12.1.21 RMU/BACKUP /NORECORD New Qualifier

A new qualifier has been added which avoids the modification of the database with recent backup information. Hence the database appears like it had not been backed up at this time.

The main purpose of this qualifier is to allow a backup of a hot standby database without modifying the database files.

Example using the /NORECORD qualifier:

```
$ RMU /BACKUP /NORECORD FOO BCK
```

12.1.22 Improved Management of the AIP (Area Inventory Page) Data by SQL Commands

Bugs 4007253, 3840715, 3019205 and 4861228

This release of Oracle Rdb changes the behavior of several DDL (data definition language) commands so that they now maintain information in the AIP (area inventory pages).

- Changed behavior for ALTER TABLE
  In prior releases of Oracle Rdb, the record length in the AIP (area inventory pages) was set when the table was created. Subsequent ALTER TABLE statements that added new columns, changed column length or data types, or dropped columns would not update this length.
  If the record length on the AIP became too out−of−date, then INSERT performance could be affected when a target page had enough room for an original row but not for the current row size. Commands such as RMU/REPAIR/INITIALIZE=LAREA_PARAMETERS/SPAM could be used to reset this length and rebuild SPAM (space management) pages that referenced the table. However, the database administrator had to calculate the revised length and be aware that the SPAM pages would need to be rebuilt for best performance.
  With this release of Oracle Rdb, the ALTER TABLE statement will track changes in the length of the table row. All such changes during a transaction are considered and, if there is an overall change in length from that currently saved in the AIP, then Rdb will update the AIP page and flag the logical area (or logical areas) so that at a convenient time the SPAM pages can be rebuilt.
  If there is no net row length change made during the transaction then no attempt is made to update the AIP or SPAM pages. Updates to the AIP are immediate since there is no SNAPSHOT support for the AIP. Therefore, these actions to update the AIP are deferred until COMMIT time.

  Note

  The record length as recorded in the AIP can change when:

  - ◊ ***A new column is added to the table (ALTER TABLE ... ADD COLUMN)***
  - ◊ ***An existing column is dropped from a table (ALTER TABLE ... DROP COLUMN)***
  - ◊ ***A data type of a column is changed to one of a different size***
  - ◊ ***The data type of the column remains the same (CHAR and VARCHAR) but the length is changed***
  - ◊ ***The RDB$FIELD_ID field increases such that the NBV (null bit vector) for the row must be expanded.***

- Changed behavior for RENAME TABLE and RENAME INDEX
  In prior releases of Oracle Rdb, the name of the logical area was not changed when a table was renamed. Apart from being confusing when trying to match the logical area names with the table names, it also caused some RMU utilities to compute incorrect values because they assumed the table name was matched by the logical area name.
  With this release of Oracle Rdb, the ALTER TABLE ... RENAME TO and RENAME TABLE statement will also revise the name of the logical area.
- Changed behavior for TRUNCATE TABLE
  Truncate table will implicitly update the AIP record length for the table. There is no need to rebuild the SPAM pages because TRUNCATE removes all rows from the table which implies there will be no SPAM references to any rows. Subsequent inserts will use the new, revised record length when searching for free space.
- Changed behavior for ALTER STORAGE MAP and ALTER INDEX
  In prior releases of Oracle Rdb, the THRESHOLDS ARE clause could not be applied to an existing

partition. Any such attempts caused an error similar to the following:

```
SQL> alter storage map sample_table_map
cont>     store using (a)
cont>           in U_EMPIDS_LOW (thresholds are (31,41,81)) with limit of (10)
cont> ;
%RDB-E-NO_META_UPDATE, metadata update failed
-RDMS-E-THRESHAREEXI, illegal thresholds usage - area U_EMPIDS_LOW exists,
and cannot have THRESHOLDS respecified
```

With this release of Oracle Rdb, the ALTER STORAGE MAP and ALTER INDEX statements will now allow this change to succeed. The new thresholds will be applied and the logical area (or logical areas) will be flagged so that at a convenient time the SPAM pages can be rebuilt.

These actions to update the AIP for length, thresholds and name are deferred until COMMIT time. Therefore, it is possible that the COMMIT following TRUNCATE TABLE may perform additional I/O if the nominal record length in the AIP differs from the actual record length of the current row version.

If there is no net row length change made during the transaction and no change of name or threshold, then Rdb will not attempt to update the AIP pages.

These problems have been corrected in Oracle Rdb Release 7.2.1.

12.1.23 New RMU Show AIP Command Added

This release of Oracle Rdb adds a new command that displays the contents of the AIP (Area Inventory Pages) structure. The AIP structure provides a mapping for logical areas to physical areas as well as describing each of those logical areas. Information such as the logical area name, length of the stored record, storage thresholds and other information can now be displayed using this simple command interface. In prior versions, the RMU Dump Larea=RDB$AIP command was the only RMU command that displayed this information.

Format

RMU/SHOW AIP rootfile [larea-name] [/LAREA=(n [,...])] [/OPTION=REBUILD_SPAMS] [/OUTPUT=output-filename] [/TYPE=type-name]

Description

The RMU Show AIP command allows the database administrator to display details of selected logical areas or all logical areas in the database.

Command Parameters

- root-file-spec
  The file specification for the database root file to be processed. The default file extension is .rdb.
- larea-name
  An optional parameter that allows the logical areas to be selected by name. Only those AIP entries are displayed. This parameter is optional and will default to all logical areas being displayed.
  Any partitioned index or table will create multiple logical areas all sharing the same name. This string

may contain standard OpenVMS file card characters (% and *) so that different names can be matched. Therefore, it is possible for many logical areas to match this name. A list of logical area names cannot be specified.
The value of ***larea−name*** may be delimited so that mixed case characters, punctuation and various character sets can be used.

Command Qualifiers

- Larea
  Specifies a list of logical area identifiers. The LAREA qualifier and larea−name parameter are mutually exclusive. The default if neither the LAREA qualifier nor the larea−name parameter is specified is to display all AIP entries.
- Output [= outout−filename]
  This qualifier is used to capture the output in a named file. If used, a standard RMU header is added to identify the command and database being processed. If omitted, the output is written to SYS$OUTPUT and no header is displayed.
- Option = REBUILD_SPAMS
  Display only those logical areas which have the REBUILD_SPAMS flag set.
- Type = type−name
  Legal values for type−name are TABLE, SORTED_INDEX, HASH_INDEX, LARGE_OBJECT, and SYSTEM_RECORD.
  This qualifier is used in conjunction with ***larea−name*** to select a subset of the AIP entries that may match a name. For instance, it is legal in Rdb to create a table and an index with the name EMPLOYEES. So using EMPLOYEES/TYPE=TABLE will make the selection unambiguous. It also allows simpler wildcarding. Commands using *EMPLOYEE*/TYPE=TABLE will process only those tables that match and not the associated index logical areas.

Usage Notes

- The database administrator requires RMU$DUMP privilege as this command is closely related to the RMU DUMP LAREA=RDB$AIP command.
- Only AIP entries that are in use are displayed. In contrast, the RMU Dump command also displays deleted and unused AIP entries.

Examples

This example uses the name of a known database table to display details for this single logical area.

Example 12−1 Displaying the AIP entry for the JOBS table

```
$ RMU/SHOW AIP SQL$DATABASE JOBS

Logical area name JOBS
Type: TABLE
Logical area 85 in mixed physical area 7
Physical area name JOBS
Record length 41
AIP page number: 151
ABM page number: 0
Snapshot Enabled TSN: 64
```

The wildcard string "*EMPLOYEE*" matches both indices and table logical areas so here we use /TYPE to limit the display to just table logical areas. The table EMPLOYEES in the MF_PERSONNEL database is partitioned across three storage areas and hence there exists three logical areas.

Example 12−2 Using wildcards and /TYPE qualifier

```
$ RMU/SHOW AIP SQL$DATABASE *EMPLOYEE*/TYPE=TABLE

Logical area name EMPLOYEES
Type: TABLE
Logical area 80 in mixed physical area 3
Physical area name EMPIDS_LOW
Record length 126
AIP page number: 150
ABM page number: 0
Snapshot Enabled TSN: 4800

Logical area name EMPLOYEES
Type: TABLE
Logical area 81 in mixed physical area 4
Physical area name EMPIDS_MID
Record length 126
AIP page number: 151
ABM page number: 0
Snapshot Enabled TSN: 1504

Logical area name EMPLOYEES
Type: TABLE
Logical area 82 in mixed physical area 5
Physical area name EMPIDS_OVER
Record length 126
AIP page number: 151
ABM page number: 0
Snapshot Enabled TSN: 1504
```

This example shows the REBUILD_SPAMS option used to locate logical areas that require SPAM rebuilds. This may occur because the stored row length changed size or THRESHOLDS were modified for the index or storage map.

Example 12−3 Locating AIP entries that need rebuilding

```
$ RMU/SHOW AIP/OPTION=REBUILD_SPAMS
_Root: SQL$DATABASE
_Logical area name:

Logical area name ACCOUNT_AUDIT
Type: TABLE
Logical area 86 in uniform physical area 1
Physical area name RDB$SYSTEM
Record length 12
Thresholds are (10, 100, 100)
Flags:
    SPAM pages should be rebuilt
AIP page number: 151
ABM page number: 1004
Snapshot Enabled TSN: 5824

Logical area name DEPARTMENTS_INDEX
```

```
Type: SORTED INDEX
Logical area 94 in uniform physical area 10
Physical area name DEPARTMENT_INFO
Record length 430
Thresholds are (30, 65, 72)
Flags:
    SPAM pages should be rebuilt
AIP page number: 151
ABM page number: 2
Snapshot Enabled TSN: 7585
```

12.1.24 New RMU Set AIP Command Added

This release of Oracle Rdb adds a new command that modifies the contents of the AIP (Area Inventory Pages) structure. The AIP structure provides a mapping for logical areas to physical areas as well describing each of those logical areas. Information such as the logical area name, length of the stored record, and storage thresholds can now be modified using this simple command interface. In prior versions, the RMU Repair Initialize=Larea_Parameters command was the only RMU command that allowed updates to this information.

Format

RMU/SET AIP root−file−spec larea−name [/LAREA=(n [, ...])] [/LENGTH[=n]] [/LOG] [/REBUILD_SPAMS] [/RENAME_TO=new−name] [/THRESHOLD=(p,q,r)]

Description

This RMU command is used to modify some attributes of an existing logical area. It cannot be used to add or delete a logical area. This command can be used to correct the record length, thresholds and name of a logical area described by an AIP entry. It can also be used to rebuild the SPAM pages for a logical area stored in UNIFORM page format areas so that threshold settings for a page correctly reflect the definition of the table.

See also the RMU Repair Spam command for information on rebuilding SPAM pages for MIXED areas.

Command Parameters

- root−file−spec
  The file specification for the database root file to be processed. The default file extension is .rdb.
- larea−name
  An optional parameter that allows the logical areas to be selected by name. Only those AIP entries are processed.
  Any partitioned index or table will create multiple logical areas all sharing the same name. This string may contain standard OpenVMS file card characters (% and *) so that different names can be matched. Therefore, it is possible for many logical areas to match this name. A list of logical area names cannot be specified.
  The value of ***larea−name*** may be delimited so that mixed case characters, punctuation and various character sets can be used.

Command Qualifiers

- Larea = (n1 [, n2 ...])
  Specifies a list of logical area identifiers. The LAREA qualifier and larea−name parameter are

mutually exclusive.

- Length [= value]
  Sets the length of the logical area. If no value is provided on the RMU Set AIP command, then Oracle Rdb will find the matching table and calculate a revised AIP nominal record length and apply it to the AIP.
- Log
  Logs the names and identifiers of logical areas modified by this command.
- Rebuild_Spams
  Locate each logical area with the "rebuild−spam" flag set and rebuild the SPAM pages.
- Rename_To = new−name
  Used to change the logical area name. This qualifier should be used with caution as some RMU commands assume a strict mapping between table/index names and names of the logical area. This command can be used to repair names that were created in older versions of Oracle Rdb where the rename table command did not propagate the change to the AIP. The value of new−name may be delimited so that mixed case, punctuation and various character sets can be used.
- Threshold = (t1 [,t2 [, t3]])
  Changes the threshold on all logical areas specified using the Larea qualifier or the larea−name parameter. RMU accepts THRESHOLD=(0,0,0) as a valid setting to disable logical area thresholds. Values must be in the range 0 through 100. Any missing values default to 100.

Usage Notes

- The database administrator requires RMU$ALTER privilege to run the command and the Rdb server also requires SELECT and ALTER privilege on the database.
- This command supersedes the RMU Repair Initialize=Larea_Parameters command that can also change the Thresholds and Length for a logical area. This command can be executed online, where as the RMU Repair command must be run offline.
- Wildcard names are not permitted with the following qualifiers to prevent accidental propagation of values to the wrong database objects.
  - ♦ LENGTH qualifier with a value is specified,
  - ♦ RENAME_TO qualifier,
  - ♦ and THRESHOLDS qualifier.
- RMU Set AIP may be used on a master database configured for HOT STANDBY. All AIP changes and SPAM rebuild actions are written to the after image journal and will be applied to the standby database. This command cannot be applied to a STANDBY database.
- THRESHOLDS for MIXED format areas are physical area attributes and are not supported at the logical area (aka AIP) level. Therefore, THRESHOLDS can not be applied to MIXED areas and specifying logical areas will cause an exception to be raised.
- The REBUILD_SPAMS qualifier is only applied to logical areas stored in UNIFORM page format storage areas.
- This command will implicitly commit any changes with no opportunity to undo them using rollback. Access to the functionality is controlled by privileges at the RMU and Rdb database level. We suggest that RMU Show AIP be used prior to any change so that you can compare the results and repeat the RMU Set AIP command with corrections if necessary.
  Some wildcard operations are restricted to prevent accidental damage to the database. For instance, a wildcard matching many objects will be rejected if more than one type of object is being changed. If a wildcard selects both table and index types, then this command will be rejected.
- This command is an online command. Each logical area will be processed within a single transaction and interact with other online users.
- When the AIP entry is changed online, any existing users of the table or index will start to use the new values if the logical areas are reloaded.

- Various SQL alter commands will register changes for the AIP and these are applied at COMMIT time. RMU Verify and RMU Show AIP Option=REBUILD_SPAMS will report any logical areas that require SPAM rebuilding. The database administrator can also examine the output from the RMU Dump Larea=RDB$AIP command.
- How long can the SPAM rebuild be delayed? The fullness of some page will have been calculated using the old AIP length or THRESHOLD values. Therefore, it might appear that a page is full when in fact the revised length will fit on the page, or the page may appear to have sufficient free space to store a row but once accessed the space is not available. By rebuilding SPAM pages, you may reduce I/O during insert operations. However, delaying the rebuild to a convenient time will not affect the integrity of the database.
- The amount of I/O required for Rebuild_Spams depends upon the number of pages allocated to the table or index involved. Assuming just one logical area is selected, then Oracle Rdb will read the ABM (Area Bitmap) to locate all SPAM pages in that area that reference this logical area. Rdb will then read each page in the SPAM interval for that SPAM page and recalculate the fullness based on the rows stored on each page.

Examples

RMU will call Rdb for each logical area that requires rebuilding.

Example 12−4 Rebuilding SPAM pages for logical areas

```
$ RMU/SET AIP/REBUILD_SPAMS MF_PERSONNEL
%RMU−I−AIPSELMOD, Logical area id 86, name ACCOUNT_AUDIT selected for
modification
%RMU−I−AIPSELMOD, Logical area id 94, name DEPARTMENTS_INDEX selected for
modification
```

RMU will request that the EMPLOYEES table length be updated in the AIP. Oracle Rdb will use the latest table layout to calculate the length in the AIP and write this back to the AIP. The EMPLOYEES table is partitioned across three storage areas and therefore the Log qualifier shows these three logical areas being updated.

Example 12−5 Updating the length in the AIP for a table

```
$ RMU/SET AIP MF_PERSONNEL EMPLOYEES/LENGTH/LOG
%RMU−I−AIPSELMOD, Logical area id 80, name EMPLOYEES selected for modification
%RMU−I−AIPSELMOD, Logical area id 81, name EMPLOYEES selected for modification
%RMU−I−AIPSELMOD, Logical area id 82, name EMPLOYEES selected for modification
```

RMU will request that the EMPLOYEES table length be updated in the AIP and then the SPAM pages will be rebuilt. This is an ONLINE operation. Note: there is an implied relationship between the logical area name and the name of the object. This example assumes that the EMPLOYEES object is mapped to a UNIFORM page format area.

Example 12−6 Updating the length for a table and rebuilding SPAM pages

```
$ RMU/SET AIP MF_PERSONNEL EMPLOYEES/LENGTH/REBUILD_SPAMS
```

When thresholds for an index are modified, they will not be effective until the SPAM pages are updated (rebuilt) to use these new values. The following example shows the index maintenance performed by SQL. The SET FLAGS command is used to display information about the change. Note that the change is applied at COMMIT time and that the SPAM rebuild is deferred until a later time. RMU Set AIP is then used to rebuild the SPAM pages.

Example 12−7 Updating the thresholds for a SORTED index

```
$ SQL$
SQL> set flags 'index_stats';
SQL> alter index candidates_sorted store in rdb$system (thresholds are (32,56,
77));
~Ai alter index "CANDIDATES_SORTED" (hashed=0, ordered=0)
~Ai larea length is 215
~As locking table "CANDIDATES" (PR -> PU)
~Ai: reads: async 0 synch 58, writes: async 8 synch 0
SQL> commit;
%RDMS−I−LOGMODVAL,     modified space management thresholds to (32%, 56%, 77%)
%RDMS−W−REBUILDSPAMS, SPAM pages should be rebuilt for logical area CANDIDATES_
SORTED
$
$ RMU/SET AIP MF_PERSONNEL CANDIDATES_SORTED/REBUILD_SPAMS/LOG
%RMU−I−AIPSELMOD, Logical area id 74, name CANDIDATES_SORTED selected for
modification
```

Chapter 13
Enhancements And Changes Provided in Oracle Rdb Release 7.2.0.2

13.1 Enhancements And Changes Provided in Oracle Rdb Release 7.2.0.2

13.1.1 Enhancements to Concurrent DDL Statements

Bug 4761143

In prior versions of Oracle Rdb, attempts to run several ALTER TABLE ... ADD CONSTRAINT commands in different sessions in parallel would either stall waiting for another transaction to finish or fail with a deadlock as shown in the following example.

```
SQL> alter table ORDER_LINES
cont>   add constraint ORDER_LINES_FK
cont>      foreign key (order_number) references orders (order_number)
cont>      not deferrable
cont> ;
%RDB-E-DEADLOCK, request failed due to resource deadlock
-RDB-E-NO_META_UPDATE, metadata update failed
-RDMS-F-DEADLOCK, deadlock on client '........ORDE'
4544524F0000001F0000000400000055
```

This behavior occurs because of Rdb's locking of the target tables to ensure consistent data in all tables. For example, for the constraint in the example to validate, there must not be another transaction deleting rows from the ORDER_LINES table. Rdb ensures this by locking the metadata for each table referenced in a constraint definition.

To provide better concurrency, this release of Oracle Rdb now allows the following statements to be used when the target table is reserved in DATA DEFINITION mode.

- ALTER TABLE can now reference the table reserved for DATA DEFINITION MODE
  The following clauses are supported:
  - ADD CONSTRAINT
  - ALTER COLUMN ... CONSTRAINT
  - ENABLE CONSTRAINT, ENABLE PRIMARY KEY, and ENABLE UNIQUE (...)
  - DROP CONSTRAINT
  - ALTER COLUMN ... NULL
  - DISABLE CONSTRAINT, DISABLE PRIMARY KEY and DISABLE UNIQUE
  - DISABLE UNIQUE (...)

  The ADD and ENABLE CONSTRAINT are best suited to concurrent execution as they may require I/O to validate the constraint.
- ALTER INDEX can now be used to build all or part of the index on a table reserved for DATA DEFINITION mode.
  The following clauses are supported:
  - BUILD PARTITION and BUILD ALL PARTITIONS
  - REBUILD PARTITION and REBUILD ALL PARTITIONS
  - TRUNCATE PARTITION and TRUNCATE ALL PARTITIONS
  - COMMENT IS clause

The BUILD and REBUILD PARTITION operators are best suited to concurrent execution as they may require I/O to construct the new index partition.

- ALTER VIEW and CREATE VIEW may now reference a table reserved for DATA DEFINITION mode.
- COMMENT ON TABLE can now reference a table reserved for DATA DEFINITION mode.
- The statement DROP CONSTRAINT can now reference a constraint on a table reserved for DATA DEFINITION mode.

Note

In prior releases, only the CREATE INDEX statement was permitted within a transaction that reserved a table in DATA DEFINITION mode.

Most ALTER TABLE clauses are now supported for tables reserved for SHARED DATA DEFINITION. The exceptions are those clauses that change the structure of the table: ADD COLUMN, DROP COLUMN and ALTER COLUMN which changes the data type.

This problem has been corrected in Oracle Rdb Release 7.2.0.2.

13.1.2 RMU Load and Unload Now Support Table and View Synonyms

Bug 4018104

This release of Oracle Rdb, 7.2.0.2, adds support for table and view synonyms for the RMU Load and RMU Unload commands. In prior releases of Rdb, the synonym name was not understood by RMU and resulted in an error, as in the following example.

```
$ SQL$
SQL> show tables
User tables in database with filename db$:personnel
     CANDIDATES
     COLLEGES
     CURRENT_INFO                    A view.
     CURRENT_JOB                     A view.
     CURRENT_SALARY                  A view.
     DEGREES
     DEPARTMENTS
     EMPLOYEES
     JOBS
     JOB_HISTORY
     RESUMES
     SALARY_HISTORY
     WORK_STATUS
     EMPS                            A synonym for table EMPLOYEES
SQL>exit
$ rmu/unload db$:personnel emps emps
%RMU-E-OUTFILDEL, Fatal error, output file deleted
-RMU-F-RELNOTFND, Relation (EMPS) not found
```

These tools now translate the synonym to the base object and process the data as though the base table had been named. This implies that the unload interchange files (.UNL) or record definition file (.RRD) that

contain the table metadata will name the base table or view and not use the synonym name. Therefore, if the metadata is used against a different database you may need to use the /MATCH_NAME qualifier to override this name during RMU Load.

Chapter 14
Enhancements And Changes Provided in Oracle Rdb Release 7.2.0.1

14.1 Enhancements And Changes Provided in Oracle Rdb Release 7.2.0.1

14.1.1 Reduced CPU Usage and Improved Performance

Several performance enhancements have been implemented in this release of Oracle Rdb. Most of these changes are either specific to applications running on I64 systems or will have a greater effect on I64 systems. These enhancements include:

- Streamlined code path of interpreted instructions
- Reduction in use of queue related PAL code and/or PAL system services
- Reduced locking for exclusive database access
- Reduced alignment faults

14.1.2 Enhancements to the ALTER TABLE ... ALTER COLUMN Clause

Bugs 2170476 and 4874525

The ALTER COLUMN clause has been enhanced with this release of Oracle Rdb and now allows columns to be altered to and from COMPUTED BY, AUTOMATIC and IDENTITY special columns.

- ALTER COLUMN may now change an AUTOMATIC column to a normal updateable base column even if there exists constraints and indices, as long as the data types are the same. In prior releases, the presence of a database wide collating sequence prevented this action.
- A non−computed base column can now be altered to be an AUTOMATIC column. The old data is retained and the column is made read−only.
- A non−computed base column can be now be altered to be a COMPUTED BY column. The old data will not be accessible (a warning is issued for interactive SQL) and references to that column will evaluate the COMPUTED BY expression. If indices or constraints reference this column, then the ALTER TABLE statement will fail.
  Note that altering the column back to a base, or automatic, column will allow older versions of the row data to be visible (any rows inserted while the column was a COMPUTED BY column will return NULL).
- The IDENTITY syntax is now supported by ALTER TABLE ... ALTER COLUMN clause.
  If the table has no existing IDENTITY column, a new sequence for the table will be created. Care must be taken to ensure that the IDENTITY will not generate existing values for the column as this would cause INSERT to fail. Use the parameters on IDENTITY to specify an appropriate START WITH value, or modify the sequence using ALTER SEQUENCE.
  If the table has an existing IDENTITY column then an error is raised.
- In some prior versions, a computed column (AUTOMATIC, IDENTITY) could be converted to a non−computed column. However, the dependency rows were never erased from the RDB$INTERRELATIONS table. This is now handled correctly. This is not a serious problem but it may cause unexpected errors when future ALTER and DROP statements are used for those referenced objects.

Please contact Oracle Support for assistance if this problem arises.

- If an IDENTITY column is converted to a base column, a COMPUTED BY column, or AUTOMATIC column, then the special sequence is automatically dropped.
- If a column has a DEFAULT (base column or AUTOMATIC UPDATE AS column) and it is converted to a COMPUTED BY, AUTOMATIC AS or an AUTOMATIC INSERT AS column, then the DEFAULT value is removed (as these types of columns are incompatible with DEFAULT).

14.1.3 /TRANSPORT Added to RMU/REPLICATE AFTER

Bug 4109344

The /TRANSPORT qualifier has been added to the RMU/REPLICATE AFTER START and CONFIGURE commands. This new qualifier allows the network transport to be specified. The valid values are "DECNET" and "TCPIP". The specified network transport is saved in the database.

In previous releases, to use TCP/IP as the network transport for Hot Standby, the system−wide logical "RDM$BIND_HOT_NETWORK_TRANSPORT" had to be defined.

For example:

```
$ RMU/REPLICATE AFTER CONFIGURE /TRANSPORT=TCPIP /STANDBY=
REMNOD::DEV:[DIR]STANDBY_DB M_TESTDB
```

14.1.4 New SHOW STATISTICS Command for Interactive SQL

This release of Oracle Rdb adds a SHOW STATISTICS command to Interactive SQL. This command displays some simple process statistics for the current process and is used primarily to compare resource usage and elapsed time for different queries.

The following example shows the output after performing a typical query.

```
SQL> select count (*)
cont>  from employees natural full outer join job_history;

         274
1 row selected
SQL> show statistics;

                  process statistics at  5−MAR−2006 05:57:48.28
         elapsed time =   0 00:00:00.16              CPU time =   0 00:00:00.05
     page fault count = 430              pages in working set = 22768
   buffered I/O count = 26                   direct I/O count = 83
      open file count = 12               file quota remaining = 7988
           locks held = 138                   locks remaining = 16776821
      CPU utilization = 31.2%              AST quota remaining = 995
SQL>
```

The statistics are reset after each execution of SHOW STATISTICS.

Chapter 15
Enhancements And Changes Provided in Oracle Rdb Release 7.2

15.1 Enhancements And Changes Provided in Oracle Rdb Release 7.2

15.1.1 Default Floating Point Format

The Intel Itanium architecture has a 64−bit virtual address model and basic system functions similar to the Alpha architecture. However, there are some implementation differences between the two platforms that might affect user−written applications.

One of the differences is the availability of hardware−supported floating−point formats. The Intel Itanium architecture implements floating−point arithmetic in hardware using the IEEE floating−point formats, including IEEE single and IEEE double. The Alpha architecture supports both IEEE and VAX floating−point formats in hardware, and OpenVMS compilers generate code using the VAX formats by default, with options (on Alpha) to use IEEE formats. Irrespective of whether it was originally written for VAX or Alpha, an OpenVMS application that uses the default VAX floating−point formats needs to produce equivalent behavior on the Intel Itanium architecture using its native IEEE formats.

- On OpenVMS VAX and OpenVMS Alpha, VAX float is the default. VAX format data is assumed and VAX floating instructions are used.
- On OpenVMS Alpha, you can specify the compiler option /FLOAT=IEEE. In this case, IEEE format data is assumed and IEEE floating instructions are used.
- On OpenVMS I64, IEEE float is the default. IEEE format data is assumed and IEEE floating instructions are used.
- On OpenVMS I64, you can specify the compiler option /FLOAT=D_FLOAT or /FLOAT=G_FLOAT.

When you compile an OpenVMS application that specifies an option to use VAX floating−point on the Intel Itanium architecture, the compiler automatically generates code for converting floating−point formats. Whenever the application performs a sequence of arithmetic operations, this code does the following:

1. Converts VAX floating−point formats to either IEEE single or IEEE double floating−point formats.
2. Performs arithmetic operations in IEEE floating−point arithmetic.
3. Converts the resulting data from IEEE formats back to VAX formats.

VAX floating−point formats have the same number of bits and precision as their equivalent IEEE floating−point formats. For most applications, the conversion process will be transparent. Note that where no arithmetic operations are performed (VAX float fetches followed by stores), conversions will not occur. The code handles such situations as moves.

In a few cases, arithmetic calculations might have different results because of the following differences between VAX and IEEE formats:

- Values of numbers represented
- Rounding rules
- Exception behavior

Matching the default of the native IA64 compilers, the Oracle Rdb and SQL precompiler default floating−point format is now IEEE. The default for the Oracle Rdb and SQL precompilers on OpenVMS

Alpha remains as VAX floating−point format. The Oracle Rdb and SQL precompilers on OpenVMS Alpha also support IEEE floating−point format as an option.

For consistent results and data content, it is important that all portions of the application utilize the same floating−point format. Oracle strongly recommends that the floating−point format is explicitly specified on compiler and pre−compiler commands.

For similar behavior for various floating−point exception conditions, Oracle recommends that customers review and consider compiler IEEE floating−point mode options. In particular, the "FAST" option may provide behavior similar to existing applications on VAX and Alpha systems.

The Oracle Rdb on−disk structures and content and data formats remain unchanged in this release.

Oracle recommends reviewing the white paper "OpenVMS Floating−point Arithmetic on the Intel Itanium Architecture" available from HP.

15.1.2 Features Not Yet Available for OpenVMS I64

The following features or capabilities or components are not currently available to run or are known to not run reliably on OpenVMS I64 with this Oracle Rdb release.

- PL/I compiler and Oracle Rdb PL/I precompilers

15.1.3 Expect Additional Memory Consumption

Due to the increased sizes of image files (especially on Integrity servers) and more aggressive buffering and caching schemes and larger I/O size defaults, you should expect to allocate additional page file quota, working set sizes and buffered I/O byte limit quota when using Oracle Rdb Release 7.2.

In particular, when running on Integrity servers, a page file quota of perhaps three times larger may be required for some applications. It is likely that buffered I/O byte limit quota usage may double when moving to Oracle Rdb Release 7.2 (as maximum I/O sizes for some operations are significantly larger than with prior Oracle Rdb Releases).

15.1.4 Handling of Initialized Overlaid Program Sections on OpenVMS I64

On Alpha and VAX systems, initializations can be done to portions of an overlaid program section. Subsequent initializations to the same portions overwrite initializations from previous modules. The last initialization performed on any byte is used as the final one of that byte for the image being linked. On I64 systems, the ELF (Executable and Linkable Format) object language does not implement the feature of the Alpha and VAX object language which allows the initialization of portions of sections. When an initialization is made, the entire section is initialized. Subsequent initializations of this section may be performed only if the non−zero portions match in value.

Any two overlaid sections are compatible if they are identical in the non−zero values. If they are not compatible, the linker issues the following error:

```
%ILINK-E-INVOVRINI, incompatible multiple initializations for overlaid section
section: <section name>
module: <module name for first overlaid section>
file: <file name for first overlaid section>
module: <module name for second overlaid section>
file: <file name for second overlaid section>
```

In the previous message, the linker lists the first module that contributes a non−zero initialization, and the first module with an incompatible initialization. Note that this is not a full list of all incompatible initializations; it is just the first one the linker encounters.

This particular symptom may be seen with applications using Oracle Rdb when multiple modules attempt to initialize handle values. Only one module may initialize any particular handle. SQL precompilers allow initialization to be controlled with the INITIALIZE_HANDLES keyword of the SQLOPTIONS qualifier.

For more detail on the handling of initialized overlaid sections, see the HP OpenVMS Version 8.2 New Features and Documentation Overview.

15.1.5 Deleted Space in Uniform Areas Not Reclaimed by Other Users

Bug 2551066

In prior releases of Oracle Rdb, when rows were deleted from a table stored in a uniform storage area, other database users would not be aware that space was made available and could extend the storage area when inserting additional rows in the table even though free space was available.

This release of Oracle Rdb introduces a mechanism that allows database users on the same cluster node to share information regarding the availability of free space. When a user chooses a location to store new rows, the location is stored in the database global section so that other users can use that location as a starting point when searching for available space. When a user deletes rows from a table, if the location of the deleted rows is closer to the beginning of the storage area than the last page used for an insert then the starting page for the next insert is updated to the location of the lowest page that had rows deleted.

15.1.6 AIP Entries Cached for Improved Performance

Whenever a table is first accessed within a database attach, Oracle Rdb must look up the description of the table. Some of the table description is stored on disk on pages called Area Inventory Pages (AIPs). These pages are linked together in a special table or "logical area" called RDB$AIP. The AIP pages are sequentially scanned each time it is necessary to find an AIP entry. If a database has many tables defined, then it could take a significant number of I/Os to locate the desired AIP entry in the RDB$AIP list. Prior to this release, the look up was repeated each time a new attach first referenced a table. Thus, applications that often attached to and detached from a database that had many tables defined in it could expend a tremendous amount of I/O constantly reloading AIP entries from disk.

This release introduces an enhancement that, for most applications, should essentially eliminate the RDB$AIP I/O. Now, the first time that a table is referenced the AIP entry is copied into an extended lock value block. (See the OpenVMS Programming Concepts Manual for more information regarding lock value blocks.) Any subsequent reference to the table will find the desired information in the lock value block and thus not need to read the entry from disk. After the most frequently accessed tables have had their AIP entries loaded into lock

value blocks, there should be little, if any, further I/O to the RDB$AIP area.

15.1.7 Improved Rollback Performance

This release of Oracle Rdb introduces additional optimizations for rolling back transactions. These improvements affect the performance of ROLLBACK statements issued by an application and also the database recovery (DBR) process. A summary of the most significant changes are listed below:

- When reading the recovery−unit journal file (RUJ), I/Os are now done using 256 block buffers instead of reading one block at a time as was done in previous versions.
- Multiple buffers are now used to read the journal. While the contents of one buffer are being processed, data is being read into the next buffer asynchronously.
- When writing to the journal, RUJ data is copied directly into the RUJ I/O buffer from the storage area data page instead of being copied into an intermediate buffer and then to the RUJ buffer.
- When reading from the journal, journal entries are processed directly from the RUJ I/O buffer instead of being copied to an intermediate buffer first.
- When rolling back a transaction, the content of the RUJ buffer is scanned to determine what data pages will be rolled back and I/Os are started to those pages immediately. That is, asynchronous prefetches (APF) are issued for pages that will be rolled back. As journal entries are processed, new prefetches are started for subsequent journal entries as soon as buffers are available. This significantly reduces the time spent waiting for I/O completion.
- In previous releases, if a process failed and a DBR was started to recover the user, the DBR would scan the journal to locate the last entry in the journal. For large transactions, the scanning operation could take a considerable amount of time. In this release, the location of the last journal entry is maintained in shared memory. Now, when a DBR process is started it can immediately locate the last entry in the journal without having to scan the journal.

15.1.8 Index Prefetching Performance Improvements

This release of Oracle Rdb introduces an optimization for queries that do index scans to fetch rows from a table. Index scans will now prefetch data pointed to by entries in the index before the application actually requests that the rows be returned. With this optimization, in many instances when an application does request the next row from a result set, the row will already be in an I/O buffer and can be immediately returned to the application.

For example, consider the following table and index definition:

```
CREATE TABLE T1 (C1 INT, C2 INT);
CREATE INDEX I1 ON T1 (C2)
```

The following query will select rows from the table based on a range of values for column C2. Oracle Rdb chooses an index scan retrieval strategy to satisfy the query.

```
SQL> SET FLAGS 'STRATEGY';
SQL> SELECT C1 FROM T1 WHERE C2 > 100 AND C2 < 900000 ORDER BY C2;
  Conjunct        Get     Retrieval by index of relation T1
    Index name  I1 [1:1]
```

When the above query executes, the index node that contains the first C2 value that is greater than 100 is fetched. Then, each entry in the index node that is greater than 100 and less than 900000 is examined and I/O

is started for each data page pointed to by each index entry. Prefetching continues for each entry in the index node until one of the following conditions is met:

- The database ASYNCH PREFETCH DEPTH IS n BUFFERS limit is reached
- The end of the current index node is encountered
- A pointer to a duplicates node is encountered
- The key with the ending scan value (in this example, 900000) is found
- A zig−zag strategy skip is requested

After all possible prefetches have been issued, the first row in the result set is returned to the application. Subsequent fetches for additional rows will find that the I/O request for a needed buffer is already in progress or may even be completed.

Each time that a new entry is requested via the index, if prefetching was stopped due to PREFETCH DEPTH being reached or a new index node being requested, prefetching will resume if that condition is satisfied.

In some applications, the performance improvements from this optimization can be very significant. Large databases that are not readily cached by existing caching products will typically see the greatest improvement in performance.

15.1.9 Performance Improvement for Query with Constant Boolean Predicates

Bug 4205719

The customer reports that the following query where the boolean condition always returns a known value of FALSE uses a full sequential retrieval and becomes very slow on a large table:

```
set flags 'strategy,detail';
select * from resumes where  1 = 2;
Tables:
  0 = RESUMES
Conjunct: 1 = 2
Get     Retrieval sequentially of relation 0:RESUMES
0 row selected
```

Although the condition was always false and 0 rows were returned, Oracle Rdb still performed a sequential table scan. In a database with about 1 million rows, this unnecessary table scan takes a lot of time.

Oracle Rdb has been changed to detect expressions of the following forms and to avoid doing index and table scans if those expressions are non−variable and evaluated as false.

```
WHERE constant−expression
WHERE other−expression AND constant−expression
WHERE constant−expression AND other−expression
```

For example:

```
WHERE 1 = 2
WHERE (1 = 2) AND (LAST_NAME > '')
WHERE (LAST_NAME > '') AND (1 = 2)
```

This does not include expressions that contain host variables, as in the examples below, because host variables are considered to be variable.

```
WHERE :HV = 1
WHERE (:HV1 = 1) AND (LAST_NAME > '')
```

This problem has been corrected in Oracle Rdb Release 7.2.

15.1.10 Index Column Group is Enabled by Default

In prior versions, Index Column Group flag was disabled by default. If this flag is enabled, the Oracle Rdb optimizer will try to find an index that has the same leading columns as the columns of Index Column Group (or Workload Column Group). If a match is found, it uses the index prefix cardinality to calculate the column duplicity and null factors which will help the optimizer to estimate solution costs and cardinalities with higher accuracy.

This flag is now enabled by default.

The following example shows flags that are set by default. This can be overridden using the *RDMS$SET_FLAGS* logical name, or the *SET FLAGS* statement in interactive and dynamic SQL.

```
SQL> show flags

Alias RDB$DBHANDLE:
Flags currently set for Oracle Rdb:
   PREFIX,WARN_DDL,INDEX_COLUMN_GROUP,MAX_SOLUTION
   ,MAX_RECURSION(100),NOBITMAPPED_SCAN
```

RMU Collect Optimizer_Statistics

Oracle strongly recommends that customers use the "RMU /COLLECT OPTIMIZER_STATISTICS" command on databases converted from prior Rdb versions.

15.1.11 No File−System Caching When Writing Database and AIJ Backup Files

It is expected that the disk−based output file from a database or after−image journal backup operation may be relatively large, sequentially accessed and not read in the near future. In order to avoid "polluting" the file system cache and to streamline file write operations, caching by the operating system is now explicitly disabled when writing these files.

15.1.12 Estimation Refinement Rules are Enabled by Default

In prior versions, index estimation was normally performed by descending to the split level in sorted indexes. For more information, please refer to the technical article entitled "Guide to Database Performance and Tuning: Predicate Estimation".

Estimation refinement rules were available to enable greater precision in estimation on indexes of *TYPE IS SORTED RANKED* and to enable estimation on hashed indexes. These rules were enabled using the

REFINE_ESTIMATES flag.

This flag is now enabled by default so that all estimation refinement rules are enabled.

The following example shows flags that are set by default. This can be overridden using the *RDMS$SET_FLAGS* logical name, or the *SET FLAGS* statement in interactive and dynamic SQL.

```
SQL> show flags

Alias RDB$DBHANDLE:
Flags currently set for Oracle Rdb:
   PREFIX,WARN_DDL,INDEX_COLUMN_GROUP,MAX_SOLUTION
   ,MAX_RECURSION(100),REFINE_ESTIMATES(127),NOBITMAPPED_SCAN
```

Notice that the *REFINE_ESTIMATES* flag has a value of 127. Please refer to the technical article above for information on the significance of this value.

The previous behavior can be obtained by setting this flag to zero or negating the flag to disable all refinement rules.

```
SQL> set flags 'refine_estimates(0)'
SQL> show flags

Alias RDB$DBHANDLE:
Flags currently set for Oracle Rdb:
   PREFIX,WARN_DDL,INDEX_COLUMN_GROUP,MAX_SOLUTION
   ,MAX_RECURSION(100),REFINE_ESTIMATES(0),NOBITMAPPED_SCAN
SQL> exit
$ define rdms$set_flags "refine_estimates(0)"
$ sql
SQL> show flags

Alias RDB$DBHANDLE:
Flags currently set for Oracle Rdb:
   PREFIX,WARN_DDL,INDEX_COLUMN_GROUP,MAX_SOLUTION
   ,MAX_RECURSION(100),REFINE_ESTIMATES(0),NOBITMAPPED_SCAN
SQL>exit
$ define rdms$set_flags "norefine_estimates"
$ sql
SQL> show flags

Alias RDB$DBHANDLE:
Flags currently set for Oracle Rdb:
   PREFIX,WARN_DDL,INDEX_COLUMN_GROUP,MAX_SOLUTION
   ,MAX_RECURSION(100),REFINE_ESTIMATES(0),NOBITMAPPED_SCAN
SQL>exit
$ sql
SQL> set flags 'norefine_estimates'
SQL> show flags

Alias RDB$DBHANDLE:
Flags currently set for Oracle Rdb:
   PREFIX,WARN_DDL,INDEX_COLUMN_GROUP,MAX_SOLUTION
   ,MAX_RECURSION(100),REFINE_ESTIMATES(0),NOBITMAPPED_SCAN
```

15.1.13 New LIMIT TO Syntax

This release of Oracle Rdb enhances the LIMIT TO clause by allowing the programmer to skip over some number of delivered rows from the query. For instance, the first row in the result set might be the column headings loaded from a CSV data source loaded by RMU/LOAD/RECORD=FORMAT=DELIMITED and as such should be ignored by queries.

Note

Oracle recommends that the values specified for skip−expression be kept small for performance reasons. These skipped rows are still fetched and processed by the query, they are just not returned to the application.

The following example shows one use of this new feature. This query returns the 100th employee from the EMPLOYEES table.

```
SQL> select last_name, first_name, employee_id
cont> from employees
cont> order by employee_id
cont> limit to 1 skip 99 rows;
 LAST_NAME        FIRST_NAME   EMPLOYEE_ID
 Herbener         James        00471
1 row selected
```

To retrieve the last row in the sorted list, the application programmer could replace the literal value with a subselect that calculates the value. This query also shows the output from the SET FLAGS command for the query strategy. Note the "Skipn" keyword that describes the new SKIP clause.

```
SQL> set flags 'strategy,detail';
SQL> select last_name, first_name, employee_id
cont> from employees
cont> order by employee_id
cont> limit to 1
cont> skip (select count(*)−1 from employees) rows;
Tables:
  0 = EMPLOYEES
  1 = EMPLOYEES
Cross block of 2 entries
  Cross block entry 1
    Aggregate: 0:COUNT (*)
    Index only retrieval of relation 1:EMPLOYEES
      Index name  EMP_EMPLOYEE_ID [0:0]
  Cross block entry 2
    Firstn: 1
    Skipn: <agg0> − 1
    Get     Retrieval by index of relation 0:EMPLOYEES
      Index name  EMP_EMPLOYEE_ID [0:0]
 LAST_NAME        FIRST_NAME   EMPLOYEE_ID
 Herbener         James        00471
1 row selected
SQL>
```

An alternative to this query would be to use ORDER ... DESC and then to use a simple LIMIT 1 ROW clause.

This query finds the statistical median salary.

```
SQL> -- select the median salary
SQL> select salary_amount
cont> from salary_history
cont> where salary_end is NULL
cont> order by salary_amount
cont> limit to 1
cont> skip (select count(*)/2
cont>       from salary_history
cont>       where salary_end is NULL);
 SALARY_AMOUNT
    $24,166.00
1 row selected
SQL>
```

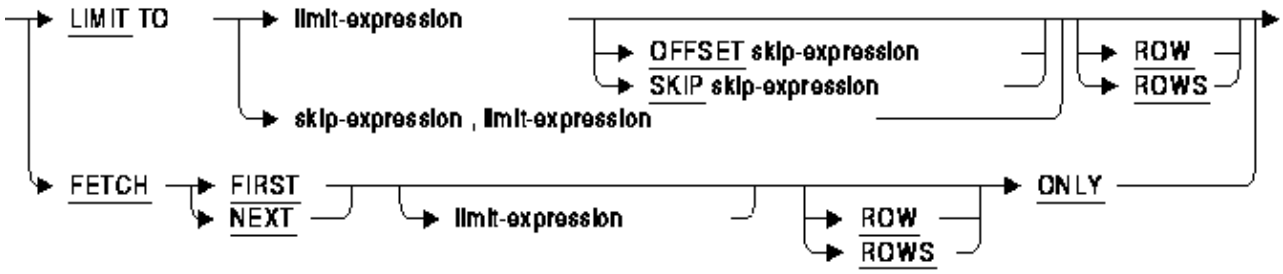
The statistical median salary can be compared with the average salary.

```
SQL> -- select the median salary compare with average
SQL> select salary_amount as median_salary,
cont>        (select avg (salary_amount)
cont>         from salary_history
cont>         where salary_end is NULL) as avg_salary edit using SALARY
cont> from salary_history
cont> where salary_end is NULL
cont> order by salary_amount
cont> limit to 1
cont> skip (select count(*)/2
cont>       from salary_history
cont>       where salary_end is NULL);
 MEDIAN_SALARY    AVG_SALARY
    $24,166.00    $31,922.79
1 row selected
SQL>
```

Syntax

The revised SQL syntax for the LIMIT TO clause is:

limit-to-clause =



The syntax variants are supported by different SQL implementations and permit different tools to use this syntax with Oracle Rdb.

Arguments

- limit−expression
  If limit−expression is evaluated to a negative value or zero then no rows are returned from the query and no error is reported.
- skip−expression
  If skip−expression is evaluated to a negative value or zero then no rows are skipped. If the skip−expression is larger than the rows in the result set then no rows are returned from the query and no error is reported.

If either limit−expression or skip−expression is specified as a numeric literal then it must be an unscaled value. These numeric expressions are converted to BIGINT before executing the query.

Neither limit−expression nor skip−expression may reference columns from the select−expression in which they occur. Only columns of a subselect specified for the limit−expression or skip−expression can be used. See above for examples that use a subselect in the LIMIT TO clause.

15.1.14 Additional %CDD−I−BLRSYNINFO Integrating with CDD

When an Oracle Rdb Release 7.2 database is integrated into a Common Data Dictionary repository, an additional %CDD−I−BLRSYNINFO is generated compared to Oracle Rdb Release 7.1. This additional message is caused by an enhancement to the RDB$DATABASE metadata table so that the value of the RDB$FILE_NAME column can be computed using a function call. The additional informational message is expected and can be ignored.

For example, executing the following SQL statements with Oracle Rdb Release 7.1 would result in a single instance of the %CDD−I−BLRSYNINFO message after the INTEGRATE statement but under Oracle Rdb Release 7.2, it results in two %CDD−I−BLRSYNINFO messages as shown.

```
SQL> CREATE DATABASE FILENAME TEMP;
SQL> DISCONNECT ALL;
SQL> INTEGRATE DATABASE FILENAME TEMP CREATE PATHNAME TEMP;
%CDD−I−BLRSYNINFO, unsupported entity − marked Incomplete
%CDD−I−BLRSYNINFO, unsupported entity − marked Incomplete
```

15.1.15 RMU Unload Record_Definition Accepts TRIM Option

This release of Oracle Rdb adds a TRIM option to the RMU Unload Record_Definition qualifier. The new TRIM option supports three keywords:

- TRAILING − trailing spaces will be trimmed from CHARACTER and CHARACTER VARYING (VARCHAR) data that is unloaded. This is the default setting if only the TRIM option is specified.
- LEADING − leading spaces will be trimmed from CHARACTER and CHARACTER VARYING (VARCHAR) data that is unloaded.
- BOTH − both leading and trailing spaces will be trimmed.

The following example shows the output without using the TRIM option.

```
$ RMU/UNLOAD/RECORD=(FORMAT=DELIMITED) DB$ WORK_STATUS SYS$OUTPUT:
```

```
"0","INACTIVE","RECORD EXPIRED"
"1","ACTIVE  ","FULL TIME     "
"2","ACTIVE  ","PART TIME     "
%RMU-I-DATRECUNL,   3 data records unloaded.
```

The results, after adding the TRIM=BOTH option, show that all trailing spaces are removed.

```
$ RMU/UNLOAD/RECORD=(FORMAT=DELIMITED,TRIM=BOTH) DB$ WORK_STATUS SYS$OUTPUT:
"0","INACTIVE","RECORD EXPIRED"
"1","ACTIVE","FULL TIME"
"2","ACTIVE","PART TIME"
%RMU-I-DATRECUNL,   3 data records unloaded.
```

15.1.16 Maximum Page and Buffer Size Increases

Previously, the maximum allowed database buffer size was 64 blocks and the maximum allowed database page size was 32 blocks. These limits have been increased. The current maximum allowed database buffer size is 128 blocks and the maximum allowed database page size is 63 blocks.

Be aware that using larger database buffer sizes will require additional virtual memory and buffered I/O byte count quota.

15.1.17 Various I/O Sizes Increased

Previously, nearly all Oracle Rdb related I/O requests were limited to a maximum of 127 blocks (65,024 bytes). In many areas within Oracle Rdb, this limit has been increased to 256 blocks (131,072 bytes). Writing to and reading from SORT work files is now done with I/O requests up to 1024 blocks (524,288 bytes). These increases should, in some cases, serve to reduce I/O counts, and increase effective I/O through−put at a lower CPU cost (by doing fewer, large I/Os).

This change also will increase the amount of virtual and physical memory required for processes in terms of page file quota and buffered I/O byte count limit.

15.1.18 New Statistics for the Oracle Rdb Executive

Bug 3917094

Several new statistics have been added to a new statistics screen for *RMU/SHOW STATISTICS*.

The following example shows the new screen.

```
Rate: 3.00 Seconds          Rdb Executive Statistics      Elapsed: 00:03:05.26
Page: 1 of 1            DUA0:[PERS.V72]MF_PERSONNEL.RDB;1         Mode: Online
------------------------------------------------------------------------------
statistic.........      rate.per.second.............. total....... average......
name..............      max..... cur..... avg....... count....... per.trans....
queries compiled              18       18        0.3           60          60.0
index scans                   13       13        0.2           42          42.0
 index only                    5        5        0.0           17          17.0
 index full                    5        5        0.0           17          17.0
dynamic optimizer              5        5        0.0           17          17.0
 one abandoned                 0        0        0.0            0           0.0
```

```
all abandoned                  0         0         0.0            0            0.0
```

In this screen, the statistics displayed have the following meanings.

- The "queries compiled" statistic counts the number of times the executive has compiled a request, also called query optimization.
- The "index scans" statistic counts the number of times an index scan is initiated. This statistic accumulates for all index types and for all scan types including direct lookup.
- The "index only" statistic counts the number of scans that are started that are index only scans.
- The "index full" statistic counts the number of index scans started that do not have a lower or upper bound. In this case, the entire index will be scanned. If a key value range is provided by the query that includes all keys in the index, the entire index will be scanned. However, this statistic will not be incremented.
- The "dynamic optimizer" statistic counts the number of times the dynamic optimizer has been invoked.
- The "one abandoned" statistic counts the number of times that the dynamic optimizer abandons a background index scan because it is considered too costly.
- The "all abandoned" statistic counts the number of times that all background indexes have been abandoned and the dynamic optimizer has switched to sequential retrieval.

15.1.19 RMU /SHOW STATISTICS Enhanced Navigation Between Row Caches

Bugs 4727723 and 3738511

Previously, when using the RMU /SHOW STATISTICS "Row Cache Utilization", "Hot Row Information", "Row Cache Status", "Row Cache Queue Length", and "Row Length Distribution" displays, it was difficult to move between displays for multiple row caches.

This problem has been corrected. The "]" and "[" keys can be used to navigate next or previous between row caches on these displays.

15.1.20 RMU SHOW LOCKS /RESOURCE_TYPE Qualifier

Previously, the RMU /SHOW LOCKS command would display all lock resource types. This could sometimes result in a significant amount of output and could make it cumbersome to locate locks for specific types of resources.

This situation has been improved with the /RESOURCE_TYPE=(restyp...) qualifier. When this qualifier is present on the command line, only the specific resource types will be displayed. This permits, for example, only PAGE or RECORD lock types to be selected. This functionality is intended primarily as a debugging tool. Knowledge of the lock types and functionality of Oracle Rdb is assumed. Not all lock types will exist on all systems and versions of Oracle Rdb.

The following keywords are allowed with the /RESOURCE_TYPE qualifier.

Table 15−1 RESOURCE_TYPE keywords

Internal Lock Type Name	Keyword(s)
ACCESS	ACCESS
ACTIVE	ACTIVE
AIJDB	AIJDB
AIJFB	AIJFB
AIJHWM	AIJHWM, AIJ_HIGH_WATER_MARK
AIJLOGMSG	AIJ_LOG_MESSAGE
AIJLOGSHIP	AIJ_LOG_SHIPPING
AIJOPEN	AIJ_OPEN
AIJSWITCH	AIJ_SWITCH
AIJ	AIJ
AIPQHD	AIP
ALS	ALS_ACTIVATION
BCKAIJ	AIJ_BACKUP, BCKAIJ
BCKAIJ_SPD	AIJ_BACKUP_SUSPEND
BUGCHK	BUGCHECK
CHAN	CHAN, FILE_CHANNEL
CLIENT	CLIENT
CLOSE	CLOSE
CLTSEQ	CLTSEQ
CPT	CORRUPT_PAGE_TABLE, CPT
DASHBOARD	DASHBOARD_NOTIFY
DBK_SCOPE	DBKEY_SCOPE
DBR	DBR_SERIALIZATION
DB	DATABASE
FIB	FAST_INCREMENTAL_BACKUP, FIB
FILID	FILID
FRZ	FREEZE
GBL_CKPT	GLOBAL_CHECKPOINT
GBPT_SLOT	GLOBAL_BPT_SLOT
KROOT	KROOT
LAREA	LAREA, LOGICAL_AREA
LOGFIL	LOGFIL
MEMBIT	MEMBIT
MONID	MONID, MONITOR_ID
MONITOR	MONITOR
NOWAIT	NOWAIT
PLN	DBKEY, RECORD, PLN
PNO	PAGE, PNO
QUIET	QUIET
RCACHE	RCACHE
RCSREQUEST	RCS_REQUEST

RCSWAITRQST	RCS_WAIT_REQUEST
REL_AREAS	RELEASE_AREAS
REL_GRIC_REQST	RELEASE_GRIC_REQUEST
RMUCLIENT	RMU_CLIENT
ROOT_AREA	DUMMY_ROOT_AREA
RO_L1	L1_SNAP_TRUNCATION
RTUPB	RTUPB
RUJBLK	RUJBLK
RW_L2	L2_SNAP_TRUNCATION
SAC	SNAP_AREA_CURSOR
SEQBLK	SEQBLK
STAREA	STORAGE_AREA, PAREA
STATRQST	STATISTICS_REQUEST
TRM	TERMINATION
TSNBLK	TSNBLK
UTILITY	UTILITY

The RESOURCE_TYPE qualifier is incompatible with the MODE, LIMIT, LOCK and PROCESS qualifiers.

15.1.21 RMU Command Qualifiers Accept Absolute or Delta Date/Time Specification

The allowed date/time format for several RMU command qualifiers have been extended to include delta as well as absolute times. The following command qualifiers now allow delta or absolute time specifications:

- RMU /SHOW STATISTICS /UNTIL=date/time
- RMU /UNLOAD /AFTER_JOURNAL /BEFORE=date/time
- RMU /UNLOAD /AFTER_JOURNAL /SINCE=date/time
- RMU /CHECKPOINT /UNTIL=date/time
- RMU /BACKUP /TAPE_EXPIRATION=date/time
- RMU /BACKUP /AFTER_JOURNAL /TAPE_EXPIRATION=date/time
- RMU /OPTIMIZE /AFTER_JOURNAL /TAPE_EXPIRATION=date/time
- RMU /BACKUP /AFTER_JOURNAL /UNTIL=date/time
- RMU /RECOVER /UNTIL=date/time
- RMU /DUMP /AFTER_JOURNAL /FIRST=TIME=date/time
- RMU /DUMP /AFTER_JOURNAL /LAST=TIME=date/time

Absolute time includes a specific date or time of day. An absolute date/time has one of the following formats:

- dd−mmm−yyyy
- hh:mm:ss.cc
- dd−mmm−yyyy:hh:mm:ss.cc
- "dd−mmm−yyyy hh:mm:ss.cc"
- BOOT
- LOGIN
- TODAY

- TOMORROW
- YESTERDAY

You can omit any of the trailing fields in the date or time. You can omit any of the fields in the middle of the format as long as you specify the punctuation marks, for example, "−mmm−yyyy hh".

Delta time is an offset from the current time to a time in the future. Delta time has the following format:

- "+[dddd−][hh:mm:ss.cc]"

You can truncate delta time after the hour field. You can also omit any of the fields after the hour field format as long as you specify the punctuation marks.

15.1.22 64−bit Statistics

In prior versions of Oracle Rdb, statistics counters were maintained in 32−bit longword integers. This limited counters to a maximum value of 4,294,967,294. This limit could be exceeded and would cause counters to "wrap" back to zero.

This problem has been corrected. Oracle Rdb statistics counters have been promoted to 64−bit quadword integers. This change effects the binary statistics output file format as well.

Note, however, that most field displays within the RMU /SHOW STATISTICS utility have not been widened and may overflow if the internal counter value exceeds the decimal display width.

15.1.23 Maximum Global Buffer Count Increased

Enhancement Bug 3820284

Prior versions of Oracle Rdb limited the total number of global buffers per database to 524,288. This limit has been relaxed. The maximum global buffer count allowed for Oracle Rdb Release 7.2 is 1,048,576.

15.1.24 Support for WORM (Write Once Read Many) Storage Removed

Prior versions of Oracle Rdb provided support for write−once storage areas on write−once, read−many (WORM) optical disk devices. This support has been removed in Oracle Rdb Release 7.2.

Databases containing storage areas configured as "WRITE ONCE" or "WORM" may not be converted to Oracle Rdb Release 7.2 format nor may they be restored using Oracle Rdb Release 7.2.

The various "WRITE ONCE" or "WORM" keywords and qualifiers in RDO, RMU and SQL are deprecated.

The SQL deprecated message is "%SQL−I−DEPR_FEATURE, Deprecated Feature: WRITE ONCE no longer supported − assuming READ WRITE attribute".

The deprecated message is generated when one specifies a clause:

- WRITE ONCE

- WRITE ONCE (JOURNAL IS ENABLED)
- WRITE ONCE (JOURNAL IS DISABLED)

within one of the following statements:

- ALTER DATABASE ... ADD STORAGE AREA
- ALTER DATABASE ... ALTER STORAGE AREA
- CREATE DATABASE ... CREATE STORAGE AREA
- IMPORT DATABASE ... CREATE STORAGE AREA

For example:

```
SQL> CREATE DATA FILENAME WORM_TEST
cont> CREATE STORAGE AREA WORM_AREA WRITE ONCE;
%SQL-I-DEPR_FEATURE, Deprecated Feature: WRITE ONCE no
longer supported - assuming READ WRITE attribute
```

Use of read−only media continues to be available. A database or storage area may be marked read−only and moved to optical media and accessed in a read−only fashion.

15.1.25 Support for ACE (AIJ Cache on Electronic disk) Removed

Prior versions of Oracle Rdb provided support for a file called an AIJ cache on an electronic disk (also known as ACE) to use as a temporary cache for AIJ write operations. At one point in time, these sorts of devices provided a performance benefit for some classes of applications that heavily used the after−image journal.

With changes in technologies (in particular, improved I/O interfaces and various write−back caching schemes), the benefits of the ACE feature have declined to the point where it is no longer an effective performance advantage. Therefore this support has been removed in Oracle Rdb Release 7.2.

The database attribute "CACHE FILENAME ..." is now ignored by Oracle Rdb. The various related keywords and qualifiers in RMU, RDO and SQL are deprecated.

15.1.26 RMU Support for /DENSITY = SDLT320

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

15.1.27 Sequential Scan Statistics

Bug 3917080

Previously, there was no way to accurately determine the number of strict sequential scans nor the number of DBKEYs returned from those sequential scans.

This problem has been corrected in Oracle Rdb Release 7.2. Two new statistics counters record the number of sequential scans started and the number of DBKEYs returned from those sequential scans. These counters are

recorded on a database−wide basis (displayed on the "Record Statistics" screen) and on a per−table basis (displayed on the "Logical Area Statistics" screens).

15.1.28 RDB$SHOVER, RDB$SETVER, SQL$SETVER Temporary Files

Previously, the RDB$SHOVER.COM, RDB$SETVER.COM and SQL$SETVER.COM procedures created temporary files when determining image file identifications. This file creation and deletion activity placed undue burden on the system and also restricted the speed of the procedures.

This problem has been corrected in Oracle Rdb Release 7.2. These procedures no longer create and delete temporary files.

15.1.29 Logical RDM$BIND_RW_TX_CHECKPOINT_ADVANCE Removed

Bug 1584167

Prior to Release 7.1.2, if the logical RDM$BIND_RW_TX_CHECKPOINT_ADVANCE was not defined to be 1, read write transactions that did not make any database modifications would not advance their fast commit checkpoint location. In Release 7.1.2, in response to Bug 2439694, the checkpointing code was restructured such that checkpoints may advance at the end of any transaction, whether or not the transactions made any database modifications. That change made the logical RDM$BIND_RW_TX_CHECKPOINT_ADVANCE no longer necessary. It has been removed.

15.1.30 Backup File Encryption

Oracle Rdb supports encryption of .RBF backup files and .AIJ after−image journal backup files using the new /ENCRYPT qualifier.

Encryption can help increase the level of security on backup data that leaves your security domain or premises. To provide a higher level of security, the backup files are always encrypted with a unique internal key. Even though you may use the same RMU command backing up the same data, the encrypted file differs from the previous backup. This is transparent to the user and the same key is used to decrypt the data.

This feature uses the OpenVMS ENCRYPT component which is included with the operating system starting with OpenVMS V8.2. All encryption algorithms supported by OpenVMS ENCRYPT can be used with RMU. Review the online help and the ENCRYPT documentation for details and supported encryption algorithms. The OpenVMS ENCRYPT component must be installed prior to using the /ENCRYPT qualifier with RMU commands.

Encryption Messages

In order to get the correct message text for encryption messages when running RMU/ENCRYPT, the following file needs to be installed using this command:

```
$INSTALL ADD SYS$MESSAGE:ENCRYPT$_MSG.EXE/OPEN/SHARED
```

The process of encryption takes readable data, called plaintext, and uses a mathematical algorithm to transform the plaintext into an unreadable, unintelligible form, called ciphertext.

To encrypt the plaintext data, the encryption operation requires a key. The key is a variable that controls the encryption operation. The same plaintext, encrypted with different keys, results in different ciphertext. In addition, repeated encryption of the same plaintext with the same key also results in different ciphertext each time.

To gain access to the data in an encrypted file, reverse the encryption process by performing the decryption process. Decryption uses a mathematical encryption algorithm to change ciphertext into the original plaintext.

You can either specify an encryption key directly or predefine a key with DCL−ENCRYPT and use the key name instead in the RMU command line.

Encryption Key

If you cannot remember the encryption key you have effectively lost all data in the encrypted file.

15.1.30.1 Commands Accepting /ENCRYPT

The "/ENCRYPT" qualifier is available for the following commands:

- RMU/BACKUP
- RMU/RESTORE
- RMU/RECOVER
- RMU/DUMP/BACKUP
- RMU/BACKUP/AFTER_JOURNAL
- RMU/DUMP/AFTER_JOURNAL
- RMU/OPTIMIZE/AFTER_JOURNAL

FORMAT=NEW_TAPE

After−image journal backup files have to be in the new tape format (/FORMAT=NEW_TAPE) in order to specify /ENCRYPT.

The /ENCRYPT qualifier has the following format: *Encrypt=([Value=|Name=][,Algorithm=])*

Table 15−2 Encrypt Keywords

Keyword	Description
NAME=key−name	Required if you do not specify key−value. Existing key name previously created and stored in the key storage table with the ENCRYPT /CREATE_KEY command. Specify either the name or the value of a key, but not both.
VALUE=key−value	Required if you do not specify key−name. Interactively defines a value for the key. Specify one of the following:

	• Character string enclosed in quotation marks (""). • 1 to 243 alphanumeric characters. Dollar signs and underscores are valid. Hexadecimal constant using the digits 0 to 9 and A to F. Specify either the name or the value of a key, but not both.
ALGORITHM=DESCBC | DESECB | DESCFB	Algorithm used to encrypt the initialization vector and the key you supply. DESCBC is the default.

Specify a key value as a string or the name of a predefined key that was created with the ENCRYPT /CREATE_KEY command. If no algorithm name is specified, the default is DESCBC. For details on the Value, Name and Algorithm parameters, review the "Encryption for OpenVMS Installation and Reference Manual".

15.1.30.2 Examples

- The following example creates a backup file which is encrypted with the specified key value string and the default encryption algorithm.

```
$ RMU/BACKUP/ENCRYPT=(VALUE="My secret key") -
    MYDB.RDB MYBACKUP.RBF
```

  This backup would be restored using a command similar to this example:

```
$ RMU/RESTORE/ENCRYPT=(VALUE="My secret key") -
    MYBACKUP.RBF
```

- The following example creates a backup file which is encrypted with the specified key name and the default encryption algorithm.

```
$ ENCRYPT /CREATE_KEY /LOG HAMLET -
    "And you yourself shall keep the key of it"
%ENCRYPT-S-KEYDEF, key defined for key name = HAMLET
$ RMU/BACKUP/ENCRYPT=NAME=HAMLET MYDB.RDB MYBACKUP.RBF
```

  This backup would be restored using a command similar to this example:

```
$ RMU/RESTORE/ENCRYPT=NAME=HAMLET MYBACKUP.RBF
```

15.1.31 RMU /POPULATE_CACHE Command /[NO]ONLY_CACHED Qualifier

The RMU /POPULATE_CACHE command allows one or more tables and indexes to be read from the database and stored in caches (if they exist). A new qualifier /[NO]ONLY_CACHED can be used to indicate that all specified tables or indexes are to be read or only those with an associated row cache.

Table 15−3 describes the command qualifiers for the RMU /POPULATE_CACHE command.

Table 15−3 RMU /POPULATE_CACHE Command Qualifiers

Qualifier	Description
/TABLE=table−list	Specifies names of one or more tables to fetch. All rows are fetched from each table. If you list multiple tables, separate the table names with a comma, and enclose the list within parentheses. Wildcard characters "*" and "%" are allowed.
/INDEX=index−list	Specifies names of one or more indexes to fetch. All nodes are fetched from each index. If you list multiple indexes, separate the index names with a comma, and enclose the list within parentheses. Wildcard characters "*" and "%" are allowed.
/LOG	Specifies whether the processing of the command is reported to SYS$OUTPUT. Specify the Log qualifier to request that information about the operation be displayed. If you specify neither /NOLOG nor /LOG, the default is the current setting of the DCL verify switch. (The DCL SET VERIFY command controls the DCL verify switch.)
/[NO]ONLY_CACHED	Specifies if table or index content is to be read only if the table or index has an associated row cache. The default is to read data only from objects that have a cache. If /NOONLY_CACHED is specified, then all data from the specified tables or indexes is read.
/TRANSACTION_TYPE=transaction_mode	Allows you to specify the transaction mode, isolation level, and wait behavior for transactions. Use one of the following keywords to control the transaction mode: • AUTOMATIC − When Transaction_Type=Automatic is specified, the transaction type depends on the current database settings for snapshots (enabled, deferred, or disabled), transaction modes available to this user, and the standby status of the database. Automatic mode is the default. • READ_ONLY − Starts read−only transactions. • WRITE − Starts read−write transactions.

15.1.32 RMU/SHOW LOCKS Includes Time and Node Name

Bug 4761828

The output of the RMU/SHOW LOCKS command has been enhanced to include the current date and time and the system node name in the header line as shown in the following example:

```
$ RMU /SHOW LOCKS
================================================================
  SHOW LOCKS Information at 26−NOV−2005 09:29:01.21 on node RDBI64
================================================================

----------------------------------------------------------------
Resource Name: AIJ journal control
Granted Lock Count: 7,  Parent Lock ID: 180007FA,   Lock Access Mode:
Executive, Resource Type: Global, Lock Value Block: 00000013 00000000
00000000 00000000
```

.
.
.

15.1.33 Default /ROW_COUNT Increased for RMU/UNLOAD and RMU/LOAD

The default value for the /ROW_COUNT qualifier for the RMU/LOAD and RMU/UNLOAD commands has been increased from 50 to 500.

The /ROW_COUNT qualifier specifies that Oracle Rdb buffer multiple rows between the Oracle Rdb server and the RMU Load process. The default is 500 rows; however, this value should be adjusted based on working set size and length of loaded data. Increasing the row count may reduce the CPU cost of the load operation. For remote databases, this may significantly reduce network traffic for large volumes of data because the buffered data can be packaged into larger network packets.

The minimum value you can specify for n is 1. The default row size is the value specified for the Commit_Every qualifier or 500, whichever is smaller.

| Contents