

# Oracle® Rdb for OpenVMS

---

## New Features Manual

Release 7.4

**April 2021**

ORACLE®

---

Oracle Rdb New Features, Release 7.4.1.1 for OpenVMS

Copyright © 1984, 2021 Oracle Corporation. **All rights reserved.**

Primary Author: Rdb Engineering and Documentation group

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

**U.S. GOVERNMENT RIGHTS** Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation shall be subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License (December 2007). Oracle America, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle, Java, Oracle Rdb, Hot Standby, LogMiner for Rdb, Oracle SQL/Services, Oracle CODASYL DBMS, Oracle RMU, Oracle CDD/Repository, Oracle Trace, and Rdb7 are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

This software or hardware and documentation may provide access to or information on content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.

---

# Contents

<b>Preface</b> .....	iii
<b>1 Enhancements And Changes Provided in Oracle Rdb Release 7.4.1.1</b>	
1.1 Enhancements .....	1-1
1.1.1 Optional Builtin Function RDB\$\$IS_ROW_FRAGMENTED .....	1-1
1.1.2 WITH READ ONLY clause for CREATE and ALTER VIEW .....	1-2
1.1.3 Enhanced conversion of date/time string literals .....	1-3
1.1.4 Support for OVERRIDING clause in INSERT and REPLACE statements .....	1-3
1.1.5 RMU RECOVER /PROGESS_REPORT Qualifier and Ctrl-T Display .....	1-5
1.1.6 Additional options for the RMU SET DATABASE command .....	1-8
1.1.7 SUMMARY_ONLY qualifier to RMU Dump Audit .....	1-9
1.1.8 New RMU VERIFY /MINIMIZE_CONFLICTS feature .....	1-10
1.1.9 New OPTION=GENERATED added to RMU Extract command .....	1-10
1.1.10 Changed behavior for the NOEDIT_FILENAME qualifier in RMU Backup After_Journal command .....	1-12
1.2 Discontinued Feature Support .....	1-12
1.2.1 Comma Statement Separator in Trigger Body No Longer Supported .....	1-12
<b>2 Enhancements And Changes Provided in Oracle Rdb Release 7.4.1.0</b>	
2.1 Enhancements .....	2-1
2.1.1 PCSI Support for Rdb Kit Installation and Deinstallation .....	2-1
2.1.2 Some Aggregate Functions Inherit Source Column EDIT STRING ...	2-2
2.1.3 Enhanced LIKE Table Support in CREATE TABLE Statement .....	2-3
2.1.4 RMU RECLAIM /FREE_PAGES Qualifier Frees Unused Data Page Clumps .....	2-6
2.1.5 CREATE DEFAULT AUDIT Supports CREATE OR REPLACE Syntax and Semantics .....	2-11
2.1.6 System Privileges Feature .....	2-12
2.1.7 Database Vault Feature .....	2-22
2.1.8 SET FLAGS Keyword for Hash Join Feature - HASHING .....	2-23
2.1.9 JOIN BY HASH Clause in CREATE OUTLINE Statement .....	2-24
2.1.10 Hash Join Feature .....	2-25
2.1.11 ALTER DATABASE ... LOAD ACL IDENTIFIERS Clause .....	2-27
2.1.12 ALTER TABLE Actions for READ ONLY Table .....	2-29
2.1.13 NULLS FIRST and NULLS LAST Options for ORDER BY Clause ..	2-30
2.1.14 Enhancements to RMU Unload After_Image (LogMiner) Interface ...	2-31
2.1.14.1 XML Option to FORMAT Qualifier .....	2-31
2.1.14.2 TRIM Option .....	2-32
2.1.14.3 SYMBOLS Qualifier .....	2-32

2.1.15	Named Partition Support for RESERVING Clause . . . . .	2-33
2.2	Discontinued Feature Support . . . . .	2-34
2.2.1	RMU Backup No Longer Supports HUFFMAN or LZSS Compression, Use ZLIB Instead . . . . .	2-34

**A Optimizer Enhancements**

A.1	Optimizer Enhancements . . . . .	A-1
A.1.1	Changes and Improvements to the Rdb Optimizer and Query Compiler . . . . .	A-1
A.1.2	Optimized NOT NULL Constraint Execution . . . . .	A-4
A.1.3	New BITMAPPED SCAN Clauses Added to OPTIMIZE Clause . . . . .	A-5
A.1.4	Query Optimization Improvements for IN Clause . . . . .	A-6
A.1.5	Query Optimization Improvements for DATE ANSI Queries . . . . .	A-8
A.1.6	New "Index Counts" Optimization for SORTED Indices . . . . .	A-8

**B RDO, RDBPRE and RDB\$INTERPRET Features**

B.1	RDO, RDBPRE and RDB\$INTERPRET Features . . . . .	B-1
B.1.1	New Request Options for RDO, RDBPRE and RDB\$INTERPRET . . . . .	B-1
B.1.2	New Language Features for RDO and Rdb Precompiler . . . . .	B-3
B.1.3	RDO Interface Now Supports Synonym References . . . . .	B-5

**Tables**

2-1	System Privileges . . . . .	2-18
-----	-----------------------------	------

---

# Preface

## Purpose of This Manual

This manual contains the New Features Chapters for Oracle Rdb Release 7.4.1.1 and prior Rdb 7.4 releases.

## Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit <http://www.oracle.com/us/support/contact/index.html> or visit <http://www.oracle.com/us/corporate/accessibility/support/index.html> if you are hearing impaired.

## Document Structure

This manual consists of the following chapters:

Chapter 1	Describes enhancements introduced in Oracle Rdb Release 7.4.1.1
Chapter 2	Describes enhancements introduced in Oracle Rdb Release 7.4.1.0
Appendix A	Describes enhancements in the Optimizer
Appendix B	Describes RDO, RDBPRE and RDB\$INTERPRET Features



# Enhancements And Changes Provided in Oracle Rdb Release 7.4.1.1

## 1.1 Enhancements

### 1.1.1 Optional Builtin Function RDB\$\$IS\_ROW\_FRAGMENTED

Oracle Rdb supports an optional builtin function that can determine if a row is fragmented. The function, RDB\$\$IS\_ROW\_FRAGMENTED must be declared as a function using the attributes and properties as shown below.

```
declare function RDB$$IS_ROW_FRAGMENTED
  (in :dbk char(8) character set unspecified)
  returns integer;
```

The following example shows the usage on the WORK\_STATUS table in the PERSONNEL database.

```
SQL> declare function RDB$$IS_ROW_FRAGMENTED
cont>   (in :dbk char(8) character set unspecified)
cont>   returns integer;
SQL>
SQL> select dbkey, RDB$$IS_ROW_FRAGMENTED (dbkey) from work_status;
          DBKEY
          99:10:12          0
          99:10:13          0
          99:10:14          0
3 rows selected
```

#### Usage Notes

- This routine may only be used from Interactive and Dynamic SQL.
- Only valid DBKEY values should be passed to the function.
- If the DBKEY passed is not the current row, then additional I/O may be required to fetch the target row.
- If the DBKEY is for a vertically partitioned table, then only the fragmented state of the primary segment is reported. There is currently no programmatic method to determine fragmented secondary segments.
- Temporary table and information table rows are never fragmented as they reside in virtual memory only.
- Fragmentation occurs when either the row is too large to fit entirely on a page or an existing row was updated to a larger size and no space existed at that time for the expanded row. The first case requires that the page size be changed for the area. However, for the second case, a DELETE and INSERT of the row might remove the fragmentation. In that case, this function allows the DBA to identify candidate fragmented rows. Fragmentation may occur when compression is enabled and the compressed row size changes due to changed data, NULL values replaced with non-NULL values, or ALTER

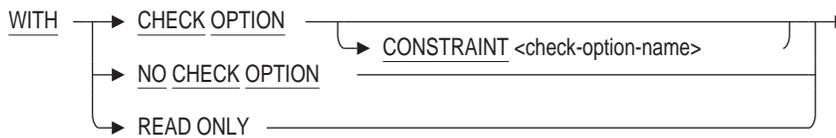
TABLE or ALTER DOMAIN statements that have increased the size of columns.

### 1.1.2 WITH READ ONLY clause for CREATE and ALTER VIEW

This release of Oracle Rdb adds support for READ ONLY view definitions, using CREATE VIEW, DECLARE LOCAL TEMPORARY VIEW or ALTER VIEW statements.

#### Syntax

check-option-clause =



Under normal circumstances, views are considered to be READ ONLY and Rdb will prevent INSERT, UPDATE and DELETE through those views when the select expression uses one of these clauses:

- \* Includes the DISTINCT operator to eliminate duplicate rows from the result table
- \* Names more than one table or view in the FROM clause
- \* Uses a derived table as the row source for a FROM clause
- \* Includes an aggregate function in the select list
- \* Contains a UNION, EXCEPT, MINUS, INTERSECT, GROUP BY, or HAVING clause

With this release the database administrator can also force a view to be READ ONLY by applying the WITH READ ONLY clause, even if the factors listed above are not true; i.e. a view that would normally be updatable is considered read-only.

This clause (WITH READ ONLY) and the WITH CHECK OPTION clause are mutually exclusive. Any CHECK OPTION constraint previously defined for the view will be deleted when WITH READ ONLY is used. Conversely if the view is altered to successfully add a CHECK OPTION then the READ ONLY attribute is removed.

The following example shows creating a view on a base table and forcing the view to be read-only.

```
SQL> create view SHOW_CURRENT_SALARY
cont> as
cont> select employee_id, salary_amount
cont> from salary_history
cont> where salary_end is null
cont> with read only
cont> ;
```

### 1.1.3 Enhanced conversion of date/time string literals

In prior releases of Oracle Rdb, string literals assigned to DATE VMS columns, or which were CAST to DATE VMS data types were translated by the SQL interface prior to being passed to the Oracle Rdb Server for execution. This small optimization could reduce or avoid the actual CAST operation.

This release of Oracle Rdb expands this support to string literals assigned to DATE ANSI, TIME, TIMESTAMP and INTERVAL data types. If the format of the string is invalid then an error is immediately reported. This is especially a benefit when applications were compiled using the SQL Precompiler or SQL Module Language which now reports the improper format during compile instead of being deferred to runtime.

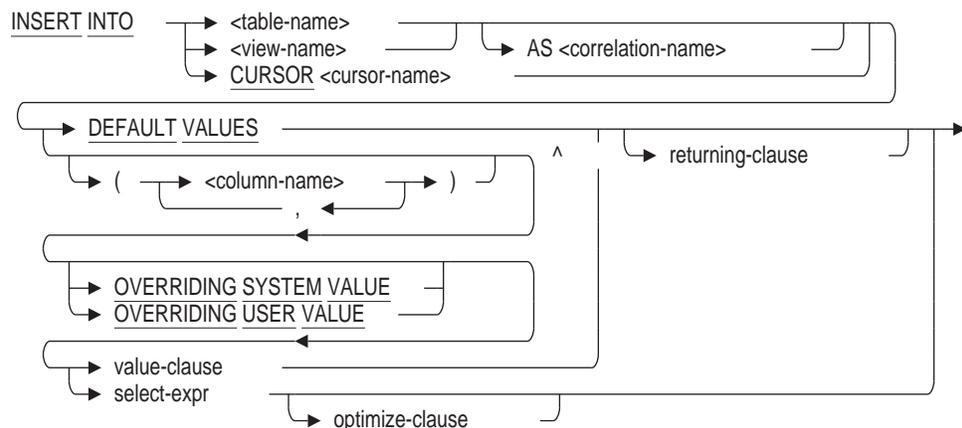
```
SQL> select cast ('1-Jan-2021' as date ansi) from rdb$database;
%SQL-F-DATCONERR, Data conversion error for string '1-Jan-2021'
-COSI-F-IVTIME, invalid date or time
SQL> select cast ('2021-1-1' as date ansi) from rdb$database;

2021-01-01
1 row selected
SQL>
```

### 1.1.4 Support for OVERRIDING clause in INSERT and REPLACE statements

This release of Oracle Rdb supports the ANSI and ISO SQL Database Language Standard OVERRIDING clause for the INSERT statement. Oracle Rdb also extends this support to the REPLACE statement. The OVERRIDING USER VALUE and OVERRIDING SYSTEM VALUE clauses affect the handling of inserts to any generated columns during INSERT or REPLACE statements. The OVERRIDING clause appears before the VALUES clause or before the SELECT clause as part of the INSERT and REPLACE statements.

#### Syntax





- \* The OVERRIDING USER VALUE clause instructs Rdb that the GENERATED, IDENTITY or AUTOMATIC AS columns will be generated by the database system and that any user supplied values will be ignored. Such a clause would be used when column names were wild carded by a SELECT clause and therefore avoids enumerating all non-generated column names.

The following example shows copying all daily sales to a log table which will generate new sales\_id values.

```
SQL> --> move daily sales and generate new column values for automatic columns
SQL> insert into SALES_LOG
cont>     overriding user value
cont>     select * from DAILY_SALES
cont> ;
```

### 1.1.5 RMU RECOVER /PROGRESS\_REPORT Qualifier and Ctrl-T Display

In this release a /PROGRESS\_REPORT qualifier has been added to the RMU RECOVER command. The /PROGRESS\_REPORT=n qualifier, where n is the time interval in seconds, displays the performance and progress of the database RMU RECOVER operation at timed intervals to SYSS\$OUTPUT.

The same display can also be output whenever Ctrl-T is typed during the RMU RECOVER operation. Ctrl-T must have been previously enabled at the DCL level using SET CONTROL=T. SET CONTROL=T requires that SET TERMINAL/BROADCAST is enabled for the display terminal.

The RMU/RECOVER performance and progress display has the following format.

- The first line is the file specification of the after image journal (AIJ) currently being recovered.
- The second line displays the number of megabytes that have been read and processed from the AIJ file during the current interval, the percentage of the AIJ file that has been processed, the number of megabytes that are currently being read per second, and the estimated completion time for the processing of this AIJ file.

The PROGRESS\_REPORT interval that has been specified in this case is 1 second.

```
DEVICE: [DIRECTORY]TEST_JOURNAL.AIJ;1
Read 21 MB (29%) at 21 MB/s, estimated completion time 11:50:29.80
```

The RMU/RECOVER performance and progress display will be different if the /FORMAT=NEW\_TAPE qualifier has been specified with the RMU/RECOVER command. This is due to limitations caused by the way AIJ data in this format is processed.

- The first line is the file specification of the current temporary AIJ work file being recovered.
- The second line displays the number of megabytes that have been read and processed from the AIJ work file during this interval, and the number of megabytes that are currently being read per second.

The `PROGRESS_REPORT` interval that has been specified in this case is 1 second.

```
DEVICE: [DIRECTORY]AIJ_WORKG8V0RS6M99D1GJKG4I80.AIJ;  
Read 36 MB at 36 MB/s
```

### Syntax

The command line syntax for the `RMU RECOVER` command `PROGRESS_REPORT` qualifier is

```
/PROGRESS_REPORT[=seconds]
```

This qualifier cannot be negated and is not the default. The default value for the `PROGRESS_REPORT` display interval is 60 seconds. The minimal value for the `PROGRESS_REPORT` display interval is 1 second.

### Examples

The following example shows the recovery of one backed up AIJ file with a progress report interval of 1 second. A zero value indicates no data was read from the AIJ file during that interval. The completion time estimate is an approximation not guaranteed to be exact.

```
$ SHOW TIME  
11-AUG-2020 16:04:39  
$ rmu/recover/LOG/root=DEVICE: [DIRECTORY]glory.rdb/PROGRESS_REPORT=1 -  
DEVICE: [DIRECTORY]backup_after.baij  
%RMU-I-LOGRECDB, recovering database file DEVICE: [DIRECTORY]GLORY.RDB;1  
%RMU-I-LOGOPNAIJ, opened journal file DEVICE: [DIRECTORY]BACKUP_AFTER.BAIJ;1  
at 11-AUG-2020 16:04:39.19  
DEVICE: [DIRECTORY]BACKUP_AFTER.BAIJ;1  
Read 10 MB ( 7%) at 10 MB/s, estimated completion time 16:04:52.75  
DEVICE: [DIRECTORY]BACKUP_AFTER.BAIJ;1  
Read 19 MB (13%) at 9 MB/s, estimated completion time 16:04:55.39  
DEVICE: [DIRECTORY]BACKUP_AFTER.BAIJ;1  
Read 52 MB (35%) at 32 MB/s, estimated completion time 16:04:45.16  
DEVICE: [DIRECTORY]BACKUP_AFTER.BAIJ;1  
Read 74 MB (49%) at 21 MB/s, estimated completion time 16:04:46.57  
DEVICE: [DIRECTORY]BACKUP_AFTER.BAIJ;1  
Read 74 MB (49%) at 0 KB/s, estimated completion time 16:04:44.19  
DEVICE: [DIRECTORY]BACKUP_AFTER.BAIJ;1  
Read 74 MB (49%) at 0 KB/s, estimated completion time 16:04:45.19  
DEVICE: [DIRECTORY]BACKUP_AFTER.BAIJ;1  
Read 74 MB (49%) at 0 KB/s, estimated completion time 16:04:46.19  
%RMU-I-LOGRECSTAT, transaction with TSN 225 committed  
DEVICE: [DIRECTORY]BACKUP_AFTER.BAIJ;1  
Read 78 MB (52%) at 3 MB/s, estimated completion time 16:05:04.80  
DEVICE: [DIRECTORY]BACKUP_AFTER.BAIJ;1  
Read 100 MB (68%) at 22 MB/s, estimated completion time 16:04:50.26  
DEVICE: [DIRECTORY]BACKUP_AFTER.BAIJ;1  
Read 118 MB (79%) at 17 MB/s, estimated completion time 16:04:50.88  
DEVICE: [DIRECTORY]BACKUP_AFTER.BAIJ;1  
Read 141 MB (95%) at 23 MB/s, estimated completion time 16:04:50.45  
DEVICE: [DIRECTORY]BACKUP_AFTER.BAIJ;1  
Read 148 MB (99%) at 6 MB/s, estimated completion time 16:04:51.20  
DEVICE: [DIRECTORY]BACKUP_AFTER.BAIJ;1  
Read 148 MB (99%) at 0 KB/s, estimated completion time 16:04:52.19  
DEVICE: [DIRECTORY]BACKUP_AFTER.BAIJ;1  
Read 148 MB (99%) at 0 KB/s, estimated completion time 16:04:53.19  
%RMU-I-LOGRECSTAT, transaction with TSN 226 committed  
%RMU-I-AIJONEDONE, AIJ file sequence 0 roll-forward operations completed  
%RMU-I-LOGRECOVR, 2 transactions committed
```

```

%RMU-I-LOGRECOVR, 0 transactions rolled back
%RMU-I-LOGRECOVR, 0 transactions ignored
%RMU-I-AIJNOACTIVE, there are no active transactions
%RMU-I-AIJSUCCESS, database recovery completed successfully
%RMU-I-AIJNXTSEQ, to continue this AIJ file recovery, the sequence number
  needed will be 1
%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-AIJSUCCESS, database recovery completed successfully
%RMU-I-AIJFNLSEQ, to start another AIJ file recovery, the sequence number
  needed will be 1
%RMU-I-AIJNOENABLED, after-image journaling has not yet been enabled
$ SHOW TIME
  11-AUG-2020 16:04:53

```

**The following example shows the recovery of two AIJ files with a progress report interval of 1 second. A zero value indicates no data was read from the AIJ file during that interval. The completion time estimate is an approximation not guaranteed to be exact.**

```

$ SHOW TIME
  11-AUG-2020 16:08:20
$ rmu/recover/root=DEVICE:[DIRECTORY]glory.rdb/PROGRESS_REPORT=1 -
DEVICE:[DIRECTORY]backup_after1.aij, -
DEVICE:[DIRECTORY]backup_after2.aij
%RMU-I-LOGRECDB, recovering database file DEVICE:[DIRECTORY]GLORY.RDB;1
%RMU-I-LOGOPNAIJ, opened journal file DEVICE:[DIRECTORY]BACKUP_AFTER1.AIJ;1
  at 11-AUG-2020 16:08:20.90
DEVICE:[DIRECTORY]BACKUP_AFTER1.AIJ;1
  Read   33 MB (44%) at   33 MB/s, estimated completion time 16:08:23.12
DEVICE:[DIRECTORY]BACKUP_AFTER1.AIJ;1
  Read   64 MB (86%) at   30 MB/s, estimated completion time 16:08:23.23
DEVICE:[DIRECTORY]BACKUP_AFTER1.AIJ;1
  Read   74 MB (99%) at    9 MB/s, estimated completion time 16:08:23.92
DEVICE:[DIRECTORY]BACKUP_AFTER1.AIJ;1
  Read   74 MB (99%) at    0 KB/s, estimated completion time 16:08:24.90
DEVICE:[DIRECTORY]BACKUP_AFTER1.AIJ;1
  Read   74 MB (99%) at    0 KB/s, estimated completion time 16:08:25.90
%RMU-I-LOGRECSTAT, transaction with TSN 225 committed
%RMU-I-LOGRECSTAT, transaction with TSN 227 committed
%RMU-I-AIJONEDONE, AIJ file sequence 0 roll-forward operations completed
%RMU-I-LOGRECOVR, 2 transactions committed
%RMU-I-LOGRECOVR, 0 transactions rolled back
%RMU-I-LOGRECOVR, 0 transactions ignored
%RMU-I-AIJACTIVE, 1 active transaction not yet committed or aborted
%RMU-I-LOGRECSTAT, transaction with TSN 226 is active
%RMU-I-AIJSUCCESS, database recovery completed successfully
%RMU-I-AIJNXTSEQ, to continue this AIJ file recovery, the sequence number
  needed will be 1
%RMU-I-LOGOPNAIJ, opened journal file DEVICE:[DIRECTORY]BACKUP_AFTER2.AIJ;1
  at 11-AUG-2020 16:08:26.63
DEVICE:[DIRECTORY]BACKUP_AFTER2.AIJ;1
  Read   32 MB (44%) at   32 MB/s, estimated completion time 16:08:28.90
DEVICE:[DIRECTORY]BACKUP_AFTER2.AIJ;1
  Read   67 MB (90%) at   34 MB/s, estimated completion time 16:08:28.82
DEVICE:[DIRECTORY]BACKUP_AFTER2.AIJ;1
  Read   73 MB (99%) at    6 MB/s, estimated completion time 16:08:29.68
DEVICE:[DIRECTORY]BACKUP_AFTER2.AIJ;1
  Read   73 MB (99%) at    0 KB/s, estimated completion time 16:08:30.63
DEVICE:[DIRECTORY]BACKUP_AFTER2.AIJ;1
  Read   73 MB (99%) at    0 KB/s, estimated completion time 16:08:31.63
%RMU-I-LOGRECSTAT, transaction with TSN 226 committed
%RMU-I-AIJONEDONE, AIJ file sequence 1 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 2
%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 2
%RMU-I-AIJNOENABLED, after-image journaling has not yet been enabled
$ SHOW TIME
  11-AUG-2020 16:08:31

```

### 1.1.6 Additional options for the RMU SET DATABASE command

This release of Oracle Rdb adds the following qualifiers to the RMU Set Database statement. This command is an OFFLINE command and requires exclusive access to the target database.

- **/NODES\_MAX**

Sets the number of nodes that are permitted to attach to the database. This command is equivalent to the SQL ALTER DATABASE ... NUMBER OF CLUSTER NODES statement.

This qualifier can only be applied to multi-file databases. RMU will report an error for single file databases. Use SQL EXPORT DATABASE and IMPORT DATABASE to change this value for a single file database.

```

$ rmu/set data/node=1 personnel
%RMU-F-MFDBONLY, operation is not allowed on single-file databases
%RMU-F-FTL_RMU, Fatal error for RMU operation at 29-OCT-2020 13:55:55.33

```

- **/RESERVE**

This clause alters the reserve limit of the database. One or more of the keywords AREAS, CACHES, JOURNALS, or SEQUENCES with new values can be specified.

This qualifier can only be applied to multi-file databases. RMU will report an error for single file databases. Use SQL EXPORT DATABASE and IMPORT DATABASE to change this value for a single file database.

```

$ rmu/set data/res=area=10 personnel
%RMU-F-MFDBONLY, operation is not allowed on single-file databases
%RMU-F-FTL_RMU, Fatal error for RMU operation at 29-OCT-2020 13:59:19.94

```

- **AREAS**

Reserves extra storage area entries to allow subsequent ALTER DATABASE ... ADD STORAGE AREA statements.

- **CACHES**

Reserves extra row cache entries to allow subsequent ALTER DATABASE ... ADD CACHE statements.

- JOURNALS

Reserves extra after image journal entries to allow subsequent ALTER DATABASE ... ADD JOURNAL statements, or RMU Set After\_Journal command.

- SEQUENCES

Reserves sequence entries. This action should be taken when a CREATE TABLE with IDENTITY or a CREATE SEQUENCE statement fails due to insufficient sequence table entries; RDMS-F-SEQTBLFUL, sequence table is full.

```
SQL> create sequence NEW_PRODUCT_CODES;
%RDB-E-NO META UPDATE, metadata update failed
-RDMS-F-SEQTBLFUL, sequence table is full
SQL>
```

- /USERS\_MAX

Sets the number of users that are permitted to attach to the database. This command is equivalent to the SQL ALTER DATABASE ... NUMBER OF USERS statement.

This qualifier can only be applied to multi-file databases. RMU will report an error for single file databases. Use SQL EXPORT DATABASE and IMPORT DATABASE to change this value for a single file database.

```
$ rmu/set data/users=11 personnel
%RMU-F-MFDBONLY, operation is not allowed on single-file databases
%RMU-F-FTL_RMU, Fatal error for RMU operation at 29-OCT-2020 13:58:29.03
```

### 1.1.7 SUMMARY\_ONLY qualifier to RMU Dump Audit

This release of Oracle Rdb adds the SUMMARY\_ONLY qualifier to RMU Dump Audit. This allows the database administrator to see a list of databases that have entries recorded in the named AUDIT\$JOURNAL.

Neither the /FORMAT nor the /TYPE qualifiers are permitted with /SUMMARY\_ONLY. The database parameter is ignored.

The following example generates a file containing the database names used by that version of the SECURITY.AUDIT\$JOURNAL.

```
$ define/nolog RMU_AJ SYS$COMMON:[SYSMGR]SECURITY.AUDIT$JOURNAL;8398
$ rmu/dump/audit -
  "" -
  RMU_AJ -
  /since=TODAY -
  /log -
  /summary_only -
  /output=audit_dump.txt
$
```

### 1.1.8 New RMU VERIFY /MINIMIZE\_CONFLICTS feature

In this release of Oracle Rdb, a /MINIMIZE\_CONFLICTS qualifier has been added to the RMU Verify command. The /MINIMIZE\_CONFLICTS qualifier attempts to reduce database page lock conflicts at timed intervals during the verify operation.

#### Syntax

- /MINIMIZE\_CONFLICTS[=n]  
This qualifier requests that RMU Verify periodically attempt to reduce locks on buffers during operation. The value of (n) is the time interval in seconds. The minimal value that can be specified is 1 second. The default time interval is 30 seconds.
- /NOMINIMIZE\_CONFLICTS  
This qualifier requests that RMU Verify request that RMU not release locking periodically. This returns RMU to prior default behavior.
- If this qualifier is omitted then the default is assumed to be MINIMIZE\_CONFLICTS=30

#### Examples

The following example shows examples for using this new MINIMIZE\_CONFLICTS feature.

```
$!  
$! Default - minimize database page lock conflicts at 30 second  
$! intervals  
$!  
$ RMU/VERIFY/ALL/NOLOG TEST.RDB  
$!  
$! Default time interval - minimize database page lock conflicts  
$! at 30 second intervals  
$!  
$ RMU/VERIFY/ALL/NOLOG/MINIMIZE_CONFLICTS TEST.RDB  
$!  
$! Minimize database page lock conflicts at 1 second intervals  
$!  
$ RMU/VERIFY/ALL/NOLOG/MINIMIZE_CONFLICTS=1 TEST.RDB  
$!  
$! Do not minimize page lock conflicts  
$!  
$ RMU/VERIFY/ALL/NOLOG/NOMINIMIZE_CONFLICTS TEST.RDB  
$
```

### 1.1.9 New OPTION=GENERATED added to RMU Extract command

This release of Oracle Rdb includes a new GENERATED option for RMU Extract. In prior releases RMU Extract ITEM=UNLOAD and ITEM=LOAD would generate load commands that assumed all the columns were updatable.

The option FULL can be used to generate syntax that loads every field by name and includes virtual columns (AUTOMATIC AS, GENERATED, IDENTITY and COMPUTED BY) as commented out names. Therefore, editing was required to uncomment GENERATED column names so they could be reloaded. In addition the /VIRTUAL=AUTOMATIC qualifier needed to be added to the RMU Load and RMU Unload commands.

Now using OPTION=(GENERATED) will instruct RMU Extract to generate more appropriate DCL commands for unloading and re-loading data in tables that contain GENERATED columns.

The following example shows a portion of a generated DCL procedure when only OPTION=FULL is used.

```
$ RMU/EXTRACT/ITEM=UNLOAD/OPTION=FULL SAMPLE_DB
.
.
.
$ CREATE SAMPLE0.COLUMNS
! Columns list for table SAMPLE0
! in ...
! Created by RMU Extract for Oracle Rdb ... on 29-JAN-2021 13:20:28.40
! Virtual: IDENT_COL
DETAILS
! Virtual: LAST_UPDATE
$ RMU/UNLOAD -
  USER1:[TESTING.DATABASES]MF_PERSONNEL_SQL.RDB -
  /FIELDS="@SAMPLE0.COLUMNS" -
  SAMPLE0 -
  SAMPLE0.UNL
$
```

The following example shows a portion of a generated DCL procedure when OPTION=(GENERATED) is used.

```
$ RMU/EXTRACT/ITEM=UNLOAD/OPTION=GENERATED SAMPLE_DB
.
.
.
$ CREATE SAMPLE0.COLUMNS
! Columns list for table SAMPLE0
! in ...
! Created by RMU Extract for Oracle Rdb ... on 29-JAN-2021 13:23:27.76
IDENT COL
DETAILS
LAST_UPDATE
$ RMU/UNLOAD -
  USER1:[TESTING.DATABASES]MF_PERSONNEL_SQL.RDB -
  /FIELDS="@SAMPLE0.COLUMNS" -
  /VIRTUAL=AUTOMATIC -
  SAMPLE0 -
  SAMPLE0.UNL
$
```

### 1.1.10 Changed behavior for the NOEDIT\_FILENAME qualifier in RMU Backup After\_Journal command

In prior releases of Oracle Rdb the /NOEDIT\_FILENAME qualifier on the RMU Backup After\_Journal was ignored. With this release it takes on a new meaning as described below:

– /EDIT\_FILENAME

As with previous versions, this qualifier defines the editing to be performed for the output backup file name. This editing is performed on the provided backup filename, or if "" is specified the default backup filename defined in the database.

This qualifier replaces any EDIT\_FILENAME defined for the database.

– /NOEDIT\_FILENAME

This qualifier negates any prior usage on the command of the /EDIT\_FILENAME qualifier and also instructs RMU to ignore the EDIT\_FILENAME defined by the SQL ALTER DATABASE statement, or RMU Set After\_Journal command. This is a change of behavior from prior versions and supports the enhancements made to the RMU Set After\_Journal command which allows the defaults to be defined for the MANUAL backup processing.

No editing is performed on the provided backup filename, or if "" is specified the default backup filename defined in the database is used without changes.

– Neither /EDIT\_FILENAME nor /NOEDIT\_FILENAME was used.

In this case RMU Backup After\_Journal will use the default if defined in the database by SQL ALTER DATABASE statement, or RMU Set After\_Journal command.

## 1.2 Discontinued Feature Support

### 1.2.1 Comma Statement Separator in Trigger Body No Longer Supported

The syntax for trigger actions in the CREATE TRIGGER statement has, in the past, supported the comma (,) as well as the semicolon (;) as statement separators. The use of the comma separator has been problematic in Oracle Rdb SQL because it conflicts in various places with the comma used as an element separator within some statements. For example, the TRACE statement allows a comma separated list of values and the INSERT INTO ... SELECT ... FROM statement allows a comma separated list of table names in the FROM clause. In these cases, a comma cannot be used as a statement separator because the current statement appears to be continued.

Future versions of Oracle Rdb are expected to include enhancements to the TRIGGER action syntax which will allow other statements to include comma as an element separator. Therefore, the comma statement separator is now no longer supported.

Any scripts or applications that include the CREATE TRIGGER statement must now be modified to use only the semicolon (;) as a separator.

This change does not affect existing database triggers, only new triggers defined using the CREATE TRIGGER statement. The RMU Extract Item=TRIGGER command already generates semicolon separators in extracted CREATE TRIGGER statements.





## Enhancements And Changes Provided in Oracle Rdb Release 7.4.1.0

### 2.1 Enhancements

#### 2.1.1 PCSI Support for Rdb Kit Installation and Deinstallation

Whenever Oracle Rdb is installed or deinstalled, Oracle Rdb will be registered in the PCSI software product database. This will allow users to use the PCSI `PRODUCT SHOW HISTORY` and `PRODUCT SHOW PRODUCT` commands to display information about releases of Oracle Rdb that have been installed or deinstalled. This information will also be helpful as input whenever a Service Request (SR) is submitted to Oracle Support.

The following lines will now be displayed during the installation of Oracle Rdb, showing that the installation has been registered in the PCSI database.

```
The following product has been selected:
  ORCL I64VMS RDB74 V7.4-100           Transition (registration)

The following product will be registered:
  ORCL I64VMS RDB74 V7.4-100           DISK$NODE84_2:[VMS$COMMON.]

File lookup pass starting ...

Portion done: 0%
...100%

File lookup pass completed search for all files listed in the product's PDF
Total files searched: 0  Files present: 0  Files absent: 0

The following product has been registered:
  ORCL I64VMS RDB74 V7.4-100           Transition (registration)
%VMSINSTAL-I-MOVEFILES, Files will now be moved to their target directories...
```

Registration in the PCSI software product database allows a user to use commands such as the following to track what Oracle Rdb releases are currently installed and the history of any past product installations and deinstallations.

```
$ PRODUCT SHOW HISTORY/SINCE
-----
PRODUCT                                KIT TYPE  OPERATION  VAL DATE
-----
ORCL I64VMS RDB74 V7.4-100             Transition Reg Product (U) 10-JUN-2020
-----

1 item found

$ PRODUCT SHOW HISTORY RDB7*
-----
PRODUCT                                KIT TYPE  OPERATION  VAL DATE
-----
ORCL I64VMS RDB74 V7.4-100             Transition Reg Product (U) 10-JUN-2020
-----

1 item found
```

```

$ PRODUCT SHOW PRODUCT RDB7*
-----
PRODUCT                                KIT TYPE    STATE
-----
ORCL I64VMS RDB74 V7.4-100            Transition  Installed
-----

```

1 item found

The following lines will now be displayed during the deinstallation of Oracle Rdb, showing that the removal of the release has been registered in the PCSI database. Deinstallation is performed by executing the DCL procedure SYSSMANAGER:RDB\$DEINSTALL\_DELETE.COM. Please refer to section "Deleting Versions of Oracle Rdb" in the Oracle Rdb Installation Guide for further details.

```

The following product has been selected:
  ORCL I64VMS RDB74 V7.4-100          Transition (registration)

The following product will be removed from destination:
  ORCL I64VMS RDB74 V7.4-100          DISK$CLYPPR84_2:[VMS$COMMON.]

Portion done: 0%...100%

```

```

The following product has been removed:
  ORCL I64VMS RDB74 V7.4-100          Transition (registration)

```

The example below shows the additional information that will be displayed by the PCSI PRODUCT commands as a result of the deinstallation of a release of Oracle Rdb.

```

$ PRODUCT SHOW HISTORY/SINCE
-----
PRODUCT                                KIT TYPE    OPERATION  VAL DATE
-----
ORCL I64VMS RDB74 V7.4-100            Transition  Remove     - 10-JUN-2020
ORCL I64VMS RDB74 V7.4-100            Transition  Reg Product (U) 10-JUN-2020
-----

```

2 items found

```

$ PRODUCT SHOW HISTORY RDB7*
-----
PRODUCT                                KIT TYPE    OPERATION  VAL DATE
-----
ORCL I64VMS RDB74 V7.4-100            Transition  Remove     - 10-JUN-2020
ORCL I64VMS RDB74 V7.4-100            Transition  Reg Product (U) 10-JUN-2020
-----

```

2 items found

```

$ PRODUCT SHOW PRODUCT RDB7*
-----
PRODUCT                                KIT TYPE    STATE
-----
0 items found

```

## 2.1.2 Some Aggregate Functions Inherit Source Column EDIT STRING

Oracle Rdb supports EDIT STRING inheritance for these functions when using Interactive SQL.

- MAX, MEDIAN, MIN, FIRST\_VALUE, LAST\_VALUE  
When the input type matches the output type, then the EDIT STRING from the source column is inherited to improve the readability of the aggregate.
- CAST  
When the datatype of the CAST includes a domain with the EDIT STRING.

The following example shows the EDIT STRING being used.

```
SQL> create domain DOM_TST integer(2) edit string '$(9)9.99';
SQL>
SQL> create table TST
cont>     (a integer(2) edit string '$(9)9.99'
cont>     ,c char(10)
cont>     );
SQL>
SQL> insert into TST
cont>     values (100, 100, 'A');
1 row inserted
SQL> insert into TST
cont>     values (233, 233, 'B');
1 row inserted
SQL>
SQL> --> column with explicit edit string
SQL> select min (a), max (a), cast (a as DOM_TST)
cont> from TST
cont> group by a
cont> ;
cont> ;

           $100.00           $100.00           $100.00
           $233.00           $233.00           $233.00
2 rows selected
SQL>
SQL> select first_value (a) within group (order by b desc),
cont>         last_value (a) within group (order by b desc),
cont>         median (a)
cont> from TST
cont> ;

           $233.00           $100.00           $166.50
1 row selected
SQL>
```

Use the SET DISPLAY NO EDIT STRING statement to disable this behavior.

### 2.1.3 Enhanced LIKE Table Support in CREATE TABLE Statement

This release of Oracle Rdb introduces support for the ANSI and ISO SQL Language Standard syntax for the LIKE table clause. It also adds new EXCLUDING and INCLUDING clauses to the LIKE clause within the CREATE TABLE statement.

In prior releases of Oracle Rdb, a table can be created using syntax similar to the following:

```
SQL> create table RETIRED_EMPLOYEES
cont>     like EMPLOYEES
cont>     ;
SQL>
```

This statement copies the definitions of each column as well as DEFAULT values defined for those source columns. SQL also allows additional columns and constraints to be defined for the new table.

```
SQL> create table RETIRED_EMPLOYEES
cont>     like EMPLOYEES
cont>     (retirement_date DATE
cont>     ,check (retirement_date > birthday) not deferrable
cont>     );
SQL>
```

This syntax is retained for backward compatibility with prior releases of Oracle Rdb.

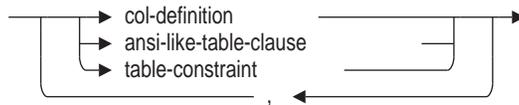
The syntax for a similar feature in the ANSI/ISO SQL Database Language moves the LIKE clause into the section that defines the columns and constraint. This adds the ability to copy column definitions from more than one table, control how GENERATED, AUTOMATIC, IDENTITY and COMPUTED columns are inherited, as well as define the column ordering; this is determined by the order of the listed columns and tables.

```
SQL> create table RETIRED_EMPLOYEES
cont>   (retirement_date DATE
cont>   ,like EMPLOYEES
cont>     including COMPUTED
cont>     excluding DEFAULTS
cont>   ,check (retirement_date > birthday) not deferrable
cont>   ,unique (employee_id)
cont>   ,hr_authorizations LIST OF BYTE VARYING
cont>   );
SQL>
```

By default, GENERATED, AUTOMATIC, IDENTITY and COMPUTED columns are not copied but columns representing the same data types are created instead.

### Syntax

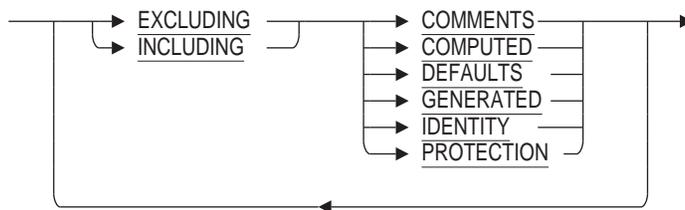
column-constraint-list =



ansi-like-table-clause =



like-attributes =



### Usage Notes

- When using the LIKE clause to copy a table definition, the creator of the new table must have REFERENCES or SELECT privilege granted for the referenced table.
- By default, Rdb includes the column protections (access control lists) and comments for any copied column. These new clauses allow the database administrator to suppress the copying of that metadata.

- The LIKE clause can be used multiple times within a CREATE TABLE statement. However, if the copied tables include any duplicate column names, then an error will be reported. Only one IDENTITY column can be defined or inherited. Use the INCLUDING IDENTITY clause, if necessary, to inherit the attributes from the referenced table.

The default behavior is EXCLUDING COMPUTED, GENERATED, IDENTITY column details. In this case, non-generated columns will be created which contain the same data type attributes. Default values defined for the source tables are not automatically inherited. Use the INCLUDING DEFAULTS clause to control this behavior.

Note: For backward compatibility with previous versions of Oracle Rdb, the LIKE clause used outside the column-constraint-list defaults to INCLUDING GENERATED, INCLUDING IDENTITY, INCLUDING COMPUTED and INCLUDING DEFAULTS. The like-attributes may not be specified in this location and therefore these defaults may not be changed.

- The clauses EXCLUDING GENERATED or INCLUDING GENERATED apply to columns defined using the GENERATED ... AS (expr) and AUTOMATIC ... AS (expr) syntax. When EXCLUDING is used or implied, the generated (or automatic) column is converted to a simple base column with the same data types.
- The clauses EXCLUDING IDENTITY or INCLUDING IDENTITY apply to columns defined using the GENERATED ... AS IDENTITY and IDENTITY (...) syntax. When EXCLUDING is used or implied, the identity column is converted to a simple base column with the same data types.
- The clauses EXCLUDING COMPUTED or INCLUDING COMPUTED apply to columns defined using the COMPUTED BY expr syntax. When EXCLUDING is used or implied, the computed by column is converted to a simple base column with the same data types. Note that the column will require space in the defined table, which isn't true for COMPUTED BY columns.
- When the LIKE clause is used within the column-constraint-list, then EXCLUDING DEFAULTS is assumed. Use the INCLUDING DEFAULTS if you wish the inherited columns to have DEFAULTS inherited from the source table.
- The LIKE clause is only used to inherit the column definitions from the referenced table. Once the table is created with LIKE clauses, subsequent changes to the source table are not propagated to the created tables.

### Examples

The following example shows the use of the LIKE clause to inherit columns from various template tables.

```

SQL> create table NAMES_REC
cont>     (LAST_NAME          LAST_NAME_DOM
cont>       ,FIRST_NAME       FIRST_NAME_DOM
cont>       ,MIDDLE_INITIAL    MIDDLE_INITIAL_DOM
cont>     );
SQL>
SQL> create table ADDRESS_REC
cont>     (ADDRESS_DATA_1     ADDRESS_DATA_1_DOM
cont>       ,ADDRESS_DATA_2     ADDRESS_DATA_2_DOM
cont>       ,CITY                CITY_DOM
cont>       ,STATE               STATE_DOM
cont>       ,POSTAL_CODE         POSTAL_CODE_DOM
cont>     );
SQL>
SQL> create table employees
cont>     (EMPLOYEE_ID         ID_DOM not null
cont>       ,like NAMES_REC     including DEFAULTS
cont>       ,like ADDRESS_REC    including DEFAULTS
cont>       ,SEX                 SEX_DOM
cont>       ,BIRTHDAY            DATE_DOM
cont>       ,STATUS_CODE         STATUS_CODE_DOM
cont>     );
SQL>

```

The resulting CREATE TABLE for the EMPLOYEES table is easier to read and allows for consistency among similar definitions.

```

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

Columns for table EMPLOYEES:
Column Name                Data Type                Domain
-----
EMPLOYEE_ID                CHAR(5)                  ID_DOM
  Not Null constraint EMPLOYEES_EMPLOYEE_ID_NOT_NULL
LAST_NAME                  CHAR(14)                 LAST_NAME_DOM
FIRST_NAME                 CHAR(10)                 FIRST_NAME_DOM
MIDDLE_INITIAL             CHAR(1)                  MIDDLE_INITIAL_DOM
ADDRESS_DATA_1             CHAR(25)                 ADDRESS_DATA_1_DOM
ADDRESS_DATA_2            CHAR(20)                 ADDRESS_DATA_2_DOM
CITY                       CHAR(20)                 CITY_DOM
STATE                      CHAR(2)                  STATE_DOM
POSTAL_CODE                CHAR(5)                  POSTAL_CODE_DOM
SEX                        CHAR(1)                  SEX_DOM
BIRTHDAY                   DATE VMS                 DATE_DOM
STATUS_CODE                CHAR(1)                  STATUS_CODE_DOM
SQL>

```

## 2.1.4 RMU RECLAIM /FREE\_PAGES Qualifier Frees Unused Data Page Clumps

There is an additional /FREE\_PAGES qualifier for the RMU Reclaim command. This qualifier is used to free unused data page clumps that are allocated in uniform storage areas. It will free all unused page clumps in an entire uniform storage area or all unused page clumps in one or more specified table or index logical areas in uniform storage areas. Any deleted dbkeys and locked space on pages will also be freed.

### Command Qualifiers

#### **/[NO]FREE\_PAGES**

Nofree\_pages is the default.

Other qualifiers may be used in conjunction with the Free\_pages qualifier.

#### **/AREA[=storage-area-name-list]**

Area is used to specify a list of uniform storage area names to process. The wild card syntax AREA=\* can be specified for processing all uniform storage areas in the database.

The default for the Area qualifier is all uniform storage areas in the database.

**/LAREA=logical-area-name-list**

Larea is used to specify a list of individual table or index logical area names to process.

There is no default for the Larea qualifier. A list of logical area names must be specified. The logical area name will be used to determine the storage area where the logical area is located. If the logical area is partitioned among multiple storage areas, each logical area partition will be processed.

This qualifier can only be specified if the Free\_pages qualifier is specified.

**/LOCK\_TIMEOUT=seconds**

Lock\_timeout is used to specify a lock timeout value that will be in effect during the execution of the RMU/RECLAIM/FREE\_PAGES command.

Lock\_timeout can only be specified if Free\_pages is also specified. The value specified with this qualifier is the maximum time in seconds during which the current RMU/RECLAIM/FREE\_PAGES command will wait to acquire an exclusive update lock on the current storage area or logical area to be processed when accessing an on-line database with other users.

If Lock\_timeout is not specified, one of the following values will be used, in the specified order of precedence.

1. The value of the logical name RDM\$BIND\_LOCK\_TIMEOUT\_INTERVAL, if it has been specified.
2. The "LOCK TIMEOUT INTERVAL" specified by the SQL CREATE or ALTER DATABASE command is used.
3. The RMU/RECLAIM/FREE\_PAGES command will wait indefinitely to acquire an exclusive update lock on the current storage area or logical area to be processed.

If /LOCK\_TIMEOUT=0 is specified, the RMU/RECLAIM/FREE\_PAGES command will ignore any lock timeout defaults that may be in effect and wait indefinitely to acquire an exclusive update lock on the current storage area or logical area to be processed.

**Usage Notes**

- The Free\_pages command can be used when the database is active. Please note that RMU will lock affected areas during processing, which may reduce concurrency.
- Free\_pages is not a default qualifier for the RMU Reclaim command. If the Free\_pages qualifier is not specified, the RMU Reclaim command will implement the default functionality of freeing deleted dbkeys and locked space in mixed and uniform database storage areas.

- RMU Reclaim Free\_Pages can be interrupted at any time; any work in progress will be rolled back. Actions that are completed will have each been committed: if processing a list of logical area names (/LAREA), a commit is performed after each logical area and if processing a list of storage areas (/AREA), a commit is performed after each storage area. Note that tables and indices which are partitioned have multiple logical areas that share the same name as the table or index.
- If Free\_pages is specified without either the Larea or Area qualifier, all the uniform storage areas in the database will be processed.
- The Area and Larea qualifiers cannot both be specified in the same RMU Reclaim command.
- If a mixed storage area name is specified with the Area qualifier or the name of a logical area in a mixed storage area is specified with the Larea qualifier, a warning message will be output and a warning status will be returned by the Reclaim command. That storage or logical area will not be processed but the Reclaim command will continue processing the next storage or logical area in the specified list of storage areas or logical areas.
- If a lock wait timeout occurs, a warning message will be output and a warning status will be returned by the Reclaim command. That storage or logical area will not be processed but the Reclaim command will continue processing the next storage area or logical area in the specified list of storage or logical areas.
- The RMU Reclaim Free\_pages functionality replaces that provided by RMU REPAIR /INITIALIZE=FREE\_PAGES. The main advantage of Reclaim is that it can be run on an active database.

## Examples

### Examples using /AREA

The following examples show the Free\_pages qualifier with the Area qualifier to free unused page clumps for one or more named storage areas.

```

$ RMU/RECLAIM/LOG/AREA=ABM_AREA1/FREE_PAGES ABM_SAMPLE.RDB
%RMU-I-RCLMAREA, Reclaiming area ABM_AREA1
%RMU-I-RCLMPAGPRC, 2138 pages processed for area ABM_AREA1
%RMU-I-RCLMPAGFREED, 1992 clump pages freed for area ABM_AREA1
$
$ RMU/RECLAIM/FREE_PAGES/AREA=(MFDBA2,MFDBA1)/LOG MFDB
%RMU-I-RCLMAREA, Reclaiming area MFDBA2
%RMU-I-RCLMPAGPRC, 13 pages processed for area MFDBA2
%RMU-I-RCLMPAGFREED, 4 clump pages freed for area MFDBA2
%RMU-I-RCLMAREA, Reclaiming area MFDBA1
%RMU-I-RCLMPAGPRC, 13 pages processed for area MFDBA1
%RMU-I-RCLMPAGFREED, 4 clump pages freed for area MFDBA1
$
$ RMU/RECLAIM/FREE_PAGES/AREA=*/LOG MFDB
%RMU-I-RCLMAREA, Reclaiming area DISK:[DIRECTORY]MFDB.RDA;1
%RMU-I-RCLMPAGPRC, 701 pages processed for area
DISK:[DIRECTORY]MFDB.RDA;1
%RMU-I-RCLMPAGFREED, 220 clump pages freed for area
DISK:[DIRECTORY]MFDB.RDA;1
%RMU-I-RCLMAREA, Reclaiming area DISK:[DIRECTORY]MFDBA1.RDA;1
%RMU-I-RCLMPAGPRC, 13 pages processed for area
DISK:[DIRECTORY]MFDBA1.RDA;1
%RMU-I-RCLMPAGFREED, 0 clump pages freed for area
DISK:[DIRECTORY]MFDBA1.RDA;1
%RMU-I-RCLMAREA, Reclaiming area DISK:[DIRECTORY]MFDBA2.RDA;1

```

```

%RMU-I-RCLMPAGPRC, 13 pages processed for area
DISK:[DIRECTORY]MFDBA2.RDA;1
%RMU-I-RCLMPAGFREED, 0 clump pages freed for area
DISK:[DIRECTORY]MFDBA2.RDA;1
%RMU-I-RCLMAREA, Reclaiming area DISK:[DIRECTORY]MFDBA3.RDA;1
%RMU-I-RCLMPAGPRC, 13 pages processed for area
DISK:[DIRECTORY]MFDBA3.RDA;1
%RMU-I-RCLMPAGFREED, 0 clump pages freed for area
DISK:[DIRECTORY]MFDBA3.RDA;1
$
$ RMU/RECLAIM/FREE_PAGES/AREA/LOG MFDB
%RMU-I-RCLMAREA, Reclaiming area DISK:[DIRECTORY]MFDB.RDA;1
%RMU-I-RCLMPAGPRC, 701 pages processed for area
DISK:[DIRECTORY]MFDB.RDA;1
%RMU-I-RCLMPAGFREED, 220 clump pages freed for area
DISK:[DIRECTORY]MFDB.RDA;1
%RMU-I-RCLMAREA, Reclaiming area DISK:[DIRECTORY]MFDBA1.RDA;1
%RMU-I-RCLMPAGPRC, 13 pages processed for area
DISK:[DIRECTORY]MFDBA1.RDA;1
%RMU-I-RCLMPAGFREED, 0 clump pages freed for area
DISK:[DIRECTORY]MFDBA1.RDA;1
%RMU-I-RCLMAREA, Reclaiming area DISK:[DIRECTORY]MFDBA2.RDA;1
%RMU-I-RCLMPAGPRC, 13 pages processed for area
DISK:[DIRECTORY]MFDBA2.RDA;1
%RMU-I-RCLMPAGFREED, 0 clump pages freed for area
DISK:[DIRECTORY]MFDBA2.RDA;1
%RMU-I-RCLMAREA, Reclaiming area DISK:[DIRECTORY]MFDBA3.RDA;1
%RMU-I-RCLMPAGPRC, 13 pages processed for area
DISK:[DIRECTORY]MFDBA3.RDA;1
%RMU-I-RCLMPAGFREED, 0 clump pages freed for area
DISK:[DIRECTORY]MFDBA3.RDA;1
$

```

### Examples using /LAREA

The following examples show the Free\_pages qualifier with the Larea qualifier to free unused page clumps for one or more named table and index logical areas.

```

$ RMU/RECLAIM/LOG/LAREA=SAMPLE_TABLE/FREE_PAGES ABM_SAMPLE.RDB
%RMU-I-RCLMLAREA, Reclaiming logical area SAMPLE_TABLE in physical
area DISK:[DIRECTORY]ABM_AREA1.RDA;1
%RMU-I-RCLMLPAGPRC, 2008 pages processed for logical area SAMPLE_TABLE
in physical area DISK:[DIRECTORY]ABM_AREA1.RDA;1
%RMU-I-RCLMLPAGFREED, 1992 clump pages freed for logical area
SAMPLE_TABLE in physical area DISK:[DIRECTORY]ABM_AREA1.RDA;1
$
$ RMU/RECLAIM/LOG/LAREA=(SAMPLE_TABLE,SAMPLE_TABLE2)/FREE_PAGES
ABM_SAMPLE.RDB
%RMU-I-RCLMLAREA, Reclaiming logical area SAMPLE_TABLE in physical
area DISK:[DIRECTORY]ABM_AREA1.RDA;1
%RMU-I-RCLMLPAGPRC, 2008 pages processed for logical area SAMPLE_TABLE
in physical area DISK:[DIRECTORY]ABM_AREA1.RDA;1
%RMU-I-RCLMLPAGFREED, 1992 clump pages freed for logical area
SAMPLE_TABLE in physical area DISK:[DIRECTORY]ABM_AREA1.RDA;1
%RMU-I-RCLMLAREA, Reclaiming logical area SAMPLE_TABLE2 in physical
area DISK:[DIRECTORY]ABM_AREA2.RDA;1
%RMU-I-RCLMLPAGPRC, 2008 pages processed for logical area
SAMPLE_TABLE2 in physical area DISK:[DIRECTORY]ABM_AREA2.RDA;1
%RMU-I-RCLMLPAGFREED, 1992 clump pages freed for logical area
SAMPLE_TABLE2 in physical area DISK:[DIRECTORY]ABM_AREA2.RDA;1
$
$ RMU/RECLAIM/LOG/LAREA=NDX_NAME/FREE_PAGES TEST_DATABASE.RDB
%RMU-I-RCLMLAREA, Reclaiming logical area NDX_NAME in
physical area DISK:[DIRECTORY]DB_DEFAULT.RDA;1
%RMU-I-RCLMLPAGPRC, 12 pages processed for logical area
NDX_NAME in physical area DISK:[DIRECTORY]DB_DEFAULT.RDA;1

```

```
%RMU-I-RCLMLPAGFREED, 3 clump pages freed for logical area
NDX_NAME in physical area DISK:[DIRECTORY]DB_DEFAULT.RDA;1
$
```

### Examples using /LAREA and partitioned index logical area

The following example shows the Free\_pages qualifier specified in an RMU Reclaim command with the Larea qualifier to free unused page clumps in an index logical area partitioned among different storage areas. Each index logical area partition is processed separately.

```
$ RMU/RECLAIM/FREE_PAGES/LOG/LAREA=INDEXA TEST_DATABASE.RDB
%RMU-I-RCLMLAREA, Reclaiming logical area INDEXA in physical area
DISK:[DIRECTORY]INDEXA_1.RDA;1
%RMU-I-RCLMLPAGPRC, 8 pages processed for logical area INDEXA in
physical area DISK:[DIRECTORY]INDEXA_1.RDA;1
%RMU-I-RCLMLPAGFREED, 3 clump pages freed for logical area INDEXA in
physical area DISK:[DIRECTORY]INDEXA_1.RDA;1
%RMU-I-RCLMLAREA, Reclaiming logical area INDEXA in physical area
DISK:[DIRECTORY]INDEXA_2.RDA;1
%RMU-I-RCLMLPAGPRC, 8 pages processed for logical area INDEXA in
physical area DISK:[DIRECTORY]INDEXA_2.RDA;1
%RMU-I-RCLMLPAGFREED, 3 clump pages freed for logical area INDEXA in
physical area DISK:[DIRECTORY]INDEXA_2.RDA;1
%RMU-I-RCLMLAREA, Reclaiming logical area INDEXA in physical area
DISK:[DIRECTORY]INDEXA_3.RDA;1
%RMU-I-RCLMLPAGPRC, 8 pages processed for logical area INDEXA in
physical area DISK:[DIRECTORY]INDEXA_3.RDA;1
%RMU-I-RCLMLPAGFREED, 3 clump pages freed for logical area INDEXA in
physical area DISK:[DIRECTORY]INDEXA_3.RDA;1
%RMU-I-RCLMLAREA, Reclaiming logical area INDEXA in physical area
DISK:[DIRECTORY]INDEXA_4.RDA;1
%RMU-I-RCLMLPAGPRC, 8 pages processed for logical area INDEXA in
physical area DISK:[DIRECTORY]INDEXA_4.RDA;1
%RMU-I-RCLMLPAGFREED, 3 clump pages freed for logical area INDEXA in
physical area DISK:[DIRECTORY]INDEXA_4.RDA;1
$
```

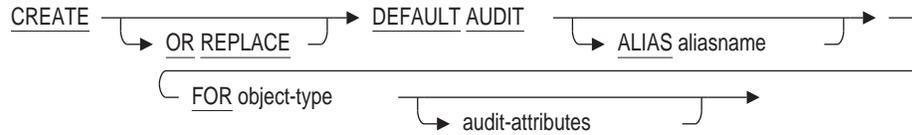
### Examples showing warnings

In the following examples, warning messages are output even if /LOG is not specified in the RMU/RECLAIM/FREE\_PAGES commands if a mixed storage area is specified or a logical area could not be processed because of a lock conflict with another user. The optional Lock\_timeout qualifier is specified.

```
$ RMU/RECLAIM/LOG/AREA=DEPARTMENTS/FREE_PAGES MF_PERSONNEL.RDB
%RMU-W-RCLMMIXIGN, Mixed area DEPARTMENTS not processed if
RMU/RECLAIM/FREE_PAGES
$
$ RMU/RECLAIM/NOLOG/AREA=ABM_AREA1/FREE_PAGES/LOCK_TIMEOUT=600 -
$ _ABM_SAMPLE.RDB
%RMU-W-RCLMARNOTPRC, Area ABM_AREA1 could not be processed due to a
lock conflict
$
$ RMU/RECLAIM/LOG/LAREA=SAMPLE_TABLE/FREE_PAGES/LOCK_TIMEOUT=1200 -
$ _ABM_SAMPLE.RDB
%RMU-I-RCLMLAREA, Reclaiming logical area SAMPLE_TABLE in physical
area DISK:[DIRECTORY]ABM_AREA1.RDA;1
%RMU-W-RCLMLARNOTPRC, Logical area SAMPLE_TABLE could not be processed
due to a lock conflict
%RMU-I-RCLMLPAGPRC, 0 pages processed for logical area SAMPLE_TABLE in
physical area DISK:[DIRECTORY]ABM_AREA1.RDA;1
%RMU-I-RCLMLPAGFREED, 0 clump pages freed for logical area
SAMPLE_TABLE in physical area DISK:[DIRECTORY]ABM_AREA1.RDA;1
$
```

## 2.1.5 CREATE DEFAULT AUDIT Supports CREATE OR REPLACE Syntax and Semantics

This release of Oracle Rdb enhances the CREATE DEFAULT AUDIT statement by allowing the OR REPLACE clause.



### Arguments

- OR REPLACE

If the OR REPLACE clause is used and the referenced object-type exists, then it will be modified using the specified audit flags and comment. Any attributes that are not specified will assume their default values.

---

#### Note

Any protections granted to the object by the GRANT statement are not replaced. They would need to be removed using the REVOKE statement.

---

If the referenced object-type does not exist, then it will be created as if the OR REPLACE clause was not used.

### Example

This example shows the CREATE DEFAULT AUDIT statement adding a new table object (which is always named RDB\$DEFAULT\_AUDIT\_TABLE).

```

SQL> create default audit
cont>   for table
cont>   all privileges
cont>   comment is 'Add a default audit table so we can inherit an ACL'
cont> ;
SQL>
SQL> grant select, delete, update, insert, show on rdb$default_audit_table to
testuser2;
SQL> grant select, dbctrl on rdb$default_audit_table to testuser3;
SQL> grant show on rdb$default_audit_table to public;
SQL>
SQL> --> display the attributes for the default table
SQL> --> note: that TESTUSER1 was the executor of the CREATE statement
SQL> show protection on table rdb$default_audit_table;
Protection on Table RDB$DEFAULT_AUDIT_TABLE
  (IDENTIFIER=[TEST,TESTUSER3],ACCESS=SELECT+DBCTRL)
  (IDENTIFIER=[TEST,TESTUSER2],ACCESS=SELECT+INSERT+UPDATE+DELETE+SHOW)
  (IDENTIFIER=[TEST,TESTUSER1],ACCESS=SELECT+INSERT+UPDATE+DELETE+SHOW+
CREATE+ALTER+DROP+DBCTRL+REFERENCES)
  (IDENTIFIER=[*,*],ACCESS=SHOW)
SQL> show audit on table rdb$default_audit_table;
Audit information for Table RDB$DEFAULT_AUDIT_TABLE
Audit Privileges:
  ALL
Alarm Privileges:
  ALL
SQL> show table (comment) rdb$default_audit_table;
Information for table RDB$DEFAULT_AUDIT_TABLE

```

Comment on table RDB\$DEFAULT\_AUDIT\_TABLE:  
Add a default audit table so we can inherit an ACL

A global temporary table.  
On commit Delete rows

SQL>

**At some later time, the table template object can be created or replaced using the CREATE OR REPLACE DEFAULT AUDIT statement.**

```
SQL> create or replace default audit
cont>     for table
cont>     type is (audit)
cont>     privileges (success, failure)
cont>     comment is 'Only audit SUCCESS and FAILURE'
cont> ;
```

SQL>

SQL> --> show that the protections are retained by OR REPLACE

SQL> show protection on table rdb\$default\_audit\_table;

```
Protection on Table RDB$DEFAULT_AUDIT_TABLE
  (IDENTIFIER=[TEST,TESTUSER3],ACCESS=SELECT+DBCTRL)
  (IDENTIFIER=[TEST,TESTUSER2],ACCESS=SELECT+INSERT+UPDATE+DELETE+SHOW)
  (IDENTIFIER=[TEST,TESTUSER1],ACCESS=SELECT+INSERT+UPDATE+DELETE+SHOW
+CREATE+ALTER+DROP+DBCTRL+REFERENCES)
  (IDENTIFIER=[*,*],ACCESS=SHOW)
```

SQL> show audit on table rdb\$default\_audit\_table;

Audit information for Table RDB\$DEFAULT\_AUDIT\_TABLE

```
Audit Privileges:
  SUCCESS,FAILURE
```

SQL> show table (comment) rdb\$default\_audit\_table;

Information for table RDB\$DEFAULT\_AUDIT\_TABLE

Comment on table RDB\$DEFAULT\_AUDIT\_TABLE:

Only audit SUCCESS and FAILURE

A global temporary table.

On commit Delete rows

SQL>

SQL> commit;

SQL>

## 2.1.6 System Privileges Feature

This release of Oracle Rdb introduces Database System Privileges - an enhancement for database security.

### Introduction

A typical Oracle Rdb database will have a security policy defined by granting privileges to users of the database and, when objects are created (tables, sequences, and so on), granting access to those objects. In addition, roles (also known as rights identifiers) may be granted specific access and any user assigned that role may inherit its access rights.

These granted privileges are stored in an access control list (ACL) with an entry for a user or role known as an access control entry (ACE).

Please refer to the SQL Reference Manual GRANT statement and REVOKE statement sections for more detailed descriptions and examples.

## Security Definitions

All objects are within the DATABASE security domain.

The primary security objects are: ASSERTION (an assertion is a form of constraint defined independently of a table definition. This feature is currently not available in SQL but maps to the standalone RDO constraints), CATALOG, COLLATING SEQUENCE, DOMAIN, OUTLINE, PROCEDURE, PROFILE, TABLE, ROLE, SCHEMA, SEQUENCE, SYNONYM, and USER.

For the purposes of system privileges, Oracle Rdb treats modules, functions and procedures as a single class - namely PROCEDURE.

Some primary security objects have an associated ACL that protects that object and controls access to sub-objects. TABLE includes the following sub-objects: VIEW, STORAGE MAP, and TRIGGER.

When TABLE, SEQUENCE, and PROCEDURE objects are created, they are implicitly given an ACL that grants ALL PRIVILEGES to the creator and NO PRIVILEGES to the PUBLIC (also known as [\*,\*]). The database administrator can override this default ACL for new primary objects:

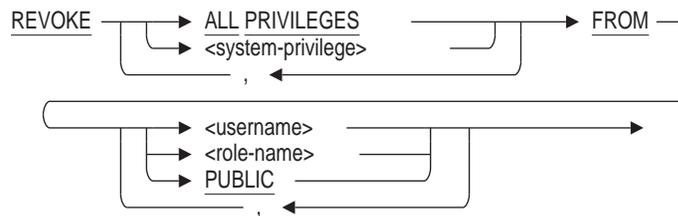
1. For tables and views, the PUBLIC access control entry will be inherited from the database access control entry for the DEFAULT user. Note: not all OpenVMS systems have a DEFAULT user defined in the system user authorization file (UAF) so that would be created by the system manager if required.

```
SQL> grant show
cont>     on database alias paysys
cont>     to default
cont> ;
SQL>
.
.
.
SQL> create table paysys.CONTROL_TABLE
cont>     (identifier_value integer generated by default as identity
cont>     );
SQL>
SQL> show protection on table paysys.CONTROL_TABLE;
Protection on Table PAYSYS.CONTROL_TABLE
  (IDENTIFIER=[ADMIN, DATABASE], ACCESS=SELECT+INSERT+UPDATE+DELETE+
  SHOW+CREATE+ALTER+DROP+DBCTRL+REFERENCES)
  (IDENTIFIER=[*,*], ACCESS=SHOW)
SQL>
```

2. Alternately, the CREATE DEFAULT AUDIT statement can be used to define template security objects in the database. These special objects are used for audit and protection inheritance. When an object (table, sequence, etc) is created, then the access control list from the security template is inherited. Note that choice (1) above will also be applied to the TABLE and VIEW default audit templates.

This example shows the **default audit** object creation and then being granted a default access for PUBLIC and specific access for users and roles.





### Arguments

- **ALL PRIVILEGES**  
All privileges will be granted (or revoked) from the listed users and roles.
- **system-privilege**  
Refer to Table 2–1, System Privileges for a list of supported system privileges.
- **TO username**  
**TO role-name**  
**TO PUBLIC**

Specifies the user name, role name, or the PUBLIC user to which you want to grant the system privilege. The PUBLIC user is the user name associated with all anonymous users who access the database.

---

#### Note

---

Oracle recommends that you only grant system privileges to trusted users. If system privileges are granted to roles then only assign those roles to trusted users.

---

If the database is defined as SECURITY CHECK IS INTERNAL and the user or role name exists as an operating system user or rights identifier, Oracle Rdb will automatically create the user or role name when you issue the GRANT statement.

- **FROM username**  
**FROM role-name**  
**FROM PUBLIC**  
Specifies the user, role, or the PUBLIC user from which the specified role is to be revoked.

### Database System Privileges

System privileges are associated with specific users (CREATE USER) and roles (CREATE ROLE) within the database. They are assigned by a user with SECURITY privilege on the database by the GRANT statement and removed by the REVOKE statement.

For example,

```

SQL> create user J_JONES identified externally;
SQL> create role DB_PROGRAMMER identified externally;
SQL>
SQL> grant create any sequence, create any procedure,
cont> create any temporary table to DB_PROGRAMMER;
SQL> grant create any trigger, create any index to J_JONES;
  
```

In this example, the database uses the default SECURITY CHECKING IS EXTERNAL therefore the granting of the rights identifier DB\_PROGRAMMER must be performed by OpenVMS. Use the AUTHORIZE utility to grant the rights identifier to specific users.

```
$ run sys$system:authorize
UAF> GRANT/ID DB_PROGRAMMER J_JONES
UAF>
```

When J\_JONES attaches to the database, they will receive the benefits of their own granted system privileges (if any) as well as those granted to the assigned role (DB\_PROGRAMMER). This is true even for DATABASE VAULT protected databases.

If a database is defined as SECURITY CHECKING IS INTERNAL, then the GRANT statement is used to associate roles created by CREATE ROLE with specific users.

```
SQL> grant DB_PROGRAMMER to J_JONES;
SQL>
```

Refer to the SQL Reference Manual GRANT Statement: Roles section for more details and examples.

### Implied System Privileges

In prior releases of Oracle Rdb, there were implied system privileges based on the database level CREATE, ALTER, DROP and SECURITY privileges that may have been granted to the user via the database ACL.

CREATE implies the permission to CREATE ANY object (except PROFILE, ROLE, and USER), and ALTER implies the permission to ALTER any object (except PROFILE, ROLE, and USER), DROP implies the permission to DROP ANY object (except PROFILE, ROLE, and USER), and SECURITY implies the permission to CREATE, ALTER, and DROP ANY PROFILE, ROLE or USER.

These implied system privileges can be displayed when attaching to a database and performing a SHOW PRIVILEGE ON DATABASE command. The implied system privileges (if any) are displayed. This is similar to the access privileges shown by this command. They reflect the ACL on the database as well as inherited access based on special privileges such as DBADM and OpenVMS privileges.

```
SQL> show privileges on database rdb$dbhandle;
Privileges on Alias RDB$DBHANDLE
      (IDENTIFIER=[RDB,RDBUSER2],ACCESS=SELECT+CREATE+ALTER+DROP)

Current system privileges:
  Granted Create Any
  COLLATING SEQUENCE, DOMAIN, ASSERTION, SESSION, OUTLINE, PROCEDURE,
  TABLE, SEQUENCE, VIEW
  Granted Alter Any
  COLLATING SEQUENCE, DOMAIN, ASSERTION, DATABASE, OUTLINE
  Granted Drop Any
  COLLATING SEQUENCE, DOMAIN, ASSERTION, DATABASE, OUTLINE
SQL>
```

Note that ALTER and DROP for primary database objects are not implicitly inherited from the database ALTER and DROP database privileges. This is because these operations are controlled by the object's own access control list.

## Fine Tuning

Although system privileges are now available in Oracle Rdb, the database administrator is not required to use them. The database will operate as it did in previous releases.

Making use of this enhanced privilege system requires that the database administrator perform these tasks.

- Use CREATE USER for each database user to which system privileges need to be granted.
- Use CREATE ROLE for any rights identifiers used to fine tune access.

Not all users and roles that access the database need to be created in the database as Rdb will still use them as in prior releases to select matching access control entries. However, users and roles must be created to allow the database administrator to grant system privileges as these objects are used to store the current privilege set.

- Use the GRANT statement to manage the CREATE ANY, ALTER ANY, DROP ANY, and TRUNCATE ANY privileges and assign them to users and roles.
- Use REVOKE on the DATABASE ALIAS to remove the CREATE, ALTER, DROP or SECURITY privileges that were previously granted to those uses. This step is required so that those database level privileges do not interfere with the fine control of system privileges.

This change will limit those users according to the new security policy. The SHOW USER and SHOW ROLE statements will display all the granted system privileges. The SHOW PRIVILEGES ON DATABASE will show the augmented system privileges based on:

- Database ACL entries,
- Granted system privileges for the attaching user,
- Granted system privileges for all rights (roles) assigned to the user,
- OpenVMS privileges (when DATABASE VAULT is enabled there will be none used)

All access control is established at ATTACH time so changes made to the USER, ROLE or DATABASE access will not have an effect until the next database ATTACH.

---

### Note

---

Oracle recommends that you only grant system privileges to trusted users. If system privileges are granted to roles then only assign those roles to trusted users.

---

**Table 2–1 System Privileges**

Operation Type	Object Type	Description
ALL PRIVILEGES		Can be used to GRANT or REVOKE all system privileges to a USER or ROLE.
ALTER ANY ...		
	ASSERTION	Permits the holder to execute the RDB CHANGE CONSTRAINT command and the SQL ALTER CONSTRAINT and COMMENT ON CONSTRAINT statements.
	CATALOG	Permits the holder to execute the ALTER CATALOG statement. The database must have multischema enabled; ALTER DATABASE ... MULTISchema IS ON;
	COLLATING SEQUENCE	Permits the holder to execute the ALTER COLLATING SEQUENCE statement.
	DOMAIN	Permits the holder to execute the ALTER DOMAIN statement.
	DATABASE	Permits the holder to execute the ALTER DATABASE and COMMENT ON DATABASE statements.
	INDEX	Permits the holder to execute the ALTER INDEX statement.
	OUTLINE	Permits the holder to execute the ALTER OUTLINE statement.
	PROCEDURE	Permits the holder to execute the ALTER FUNCTION, ALTER MODULE and ALTER PROCEDURE statements.
	PROFILE	Permits the holder to execute the ALTER PROFILE and ALTER DEFAULT PROFILE statements. When the profile exists, the holder can also execute the CREATE OR REPLACE PROFILE and CREATE OR REPLACE DEFAULT PROFILE statements.
	ROLE	Permits the holder to execute the ALTER ROLE statement.
	SEQUENCE	Permits the holder to execute the ALTER SEQUENCE statement. When the sequence exists, the holder can also execute the CREATE OR REPLACE SEQUENCE statement.
	SCHEMA	Permits the holder to execute the ALTER SCHEMA statement. The database must have multischema enabled: ALTER DATABASE ... MULTISchema IS ON;
	STORAGE MAP	Permits the holder to execute the ALTER STORAGE MAP statement.

(continued on next page)

**Table 2–1 (Cont.) System Privileges**

Operation Type	Object Type	Description
	SYNONYM	Permits the holder to execute the ALTER SYNONYM statement. When the synonym exists, the holder can also execute the CREATE OR REPLACE SYNONYM statements. The database must have synonyms enabled: ALTER DATABASE ... SYNONYMS ARE ENABLED;
	TABLE	Permits the holder to execute the ALTER TABLE statement.
	TEMPORARY TABLE	Permits the holder to execute the ALTER TEMPORARY TABLE statement or CREATE INFORMATION TABLE statement when CREATE ANY TABLE privilege is not granted.
	TRIGGER	Permits the holder to execute the ALTER TRIGGER statement.
	USER	Permits the holder to execute the ALTER USER statement.
	VIEW	Permits the holder to execute the ALTER VIEW statement. When the view exists, the holder can also execute the CREATE OR REPLACE VIEW statement.
CREATE ...	SESSION	Permits the holder to execute ATTACH, CONNECT, DECLARE ALIAS, SET SESSION AUTHORIZATION, and other session starting statements.
CREATE ANY ...	ASSERTION	Permits the holder to execute the RDO DEFINE CONSTRAINT command. The SQL equivalent to DEFINE CONSTRAINT would be a table level constraint. Such definitions are managed by TABLE privileges, therefore this privilege does not apply to SQL.
	CATALOG	Permits the holder to execute the CREATE CATALOG statement. The database must have multiscHEMA enabled: ALTER DATABASE ... MULTISCHEMA IS ON;
	COLLATING SEQUENCE	Permits the holder to execute the CREATE COLLATING SEQUENCE statement.
	DOMAIN	Permits the holder to execute the CREATE DOMAIN statement.
	INDEX	Permits the holder to execute the CREATE INDEX statement.
	OUTLINE	Permits the holder to execute the CREATE OUTLINE statement.

(continued on next page)

**Table 2–1 (Cont.) System Privileges**

Operation Type	Object Type	Description
	PROCEDURE	Permits the holder to execute the CREATE FUNCTION, CREATE MODULE and CREATE PROCEDURE statements.
	PROFILE	Permits the holder to execute the CREATE PROFILE and CREATE DEFAULT PROFILE statements.
	ROLE	Permits the holder to execute the CREATE ROLE statement.
	SEQUENCE	Permits the holder to execute the CREATE SEQUENCE statement.
	SCHEMA	Permits the holder to execute the CREATE SCHEMA statement. The database must have multiscHEMA enabled: ALTER DATABASE ... MULTISCHEMA IS ON;
	STORAGE MAP	Permits the holder to execute the CREATE STORAGE MAP statement.
	SYNONYM	Permits the holder to execute the CREATE SYNONYM statement. The database must have synonyms enabled: ALTER DATABASE ... SYNONYMS ARE ENABLED;
	TABLE	Permits the holder to execute the CREATE TABLE statement.
	TEMPORARY TABLE	Permits the holder to execute the CREATE TEMPORARY TABLE and CREATE INFORMATION TABLE statements when CREATE ANY TABLE privilege is not granted.
	TRIGGER	Permits the holder to execute the CREATE TRIGGER statement.
	USER	Permits the holder to execute the CREATE USER statement.
	VIEW	Permits the holder to execute the CREATE VIEW statement.
DROP ANY ...		
	ASSERTION	Permits the holder to execute the RDO DELETE CONSTRAINT command or the SQL DROP CONSTRAINT statement.
	CATALOG	Permits the holder to execute the DROP CATALOG statement. The database must have multiscHEMA enabled: ALTER DATABASE ... MULTISCHEMA IS ON;
	COLLATING SEQUENCE	Permits the holder to execute the DROP COLLATING SEQUENCE statement.
	DOMAIN	Permits the holder to execute the DROP DOMAIN statement.
	DATABASE	Permits the holder to execute the DROP DATABASE statement.

(continued on next page)

**Table 2–1 (Cont.) System Privileges**

Operation Type	Object Type	Description
	INDEX	Permits the holder to execute the DROP INDEX statement.
	OUTLINE	Permits the holder to execute the DROP OUTLINE statement.
	PROCEDURE	Permits the holder to execute the DROP FUNCTION, DROP MODULE and DROP PROCEDURE statements.
	PROFILE	Permits the holder to execute the DROP PROFILE and DROP DEFAULT PROFILE statements.
	ROLE	Permits the holder to execute the DROP ROLE statement.
	SEQUENCE	Permits the holder to execute the DROP SEQUENCE statement.
	SCHEMA	Permits the holder to execute the DROP SCHEMA statement. The database must have multischema enabled: ALTER DATABASE ... MULTISCHHEMA IS ON;
	STORAGE MAP	Permits the holder to execute the DROP STORAGE MAP statement.
	SYNONYM	Permits the holder to execute the DROP SYNONYM statement. The database must have synonyms enabled: ALTER DATABASE ... SYNONYMS ARE ENABLED;
	TABLE	Permits the holder to execute the DROP TABLE statement.
	TEMPORARY TABLE	Permits the holder to execute the DROP TEMPORARY TABLE statement or DROP INFORMATION TABLE statement when DROP ANY TABLE privilege is not granted.
	TRIGGER	Permits the holder to execute the DROP TRIGGER statement.
	USER	Permits the holder to execute the DROP USER statement.
	VIEW	Permits the holder to execute the DROP VIEW statement.
TRUNCATE ...	ANY TABLE	Permits the holder to execute the TRUNCATE TABLE statement. This privilege effectively allows the user to temporarily disable triggers during the TRUNCATE operation and assume DELETE access to the table.
	TABLE	This privilege is similar to the TRUNCATE ANY TABLE privilege but requires that the user also be granted DELETE access to the table. Permits the user to execute the TRUNCATE TABLE statement without further privilege checking for BEFORE and AFTER DELETE triggers.

## Usage Notes

- You must have the SECURITY privilege on the database to grant a system privilege to a user or a role.
- You must have the SECURITY privilege on the database to revoke a system privilege from a user or a role.
- The TEMPORARY TABLE class of privileges is considered a subset of TABLE and allows the database administrator to grant privileges to a user but only for logical tables not physical (base) tables.
- If the user has CREATE ANY TABLE, then the CREATE ANY TEMPORARY TABLE privilege is not required. Similarly, ALTER ANY TEMPORARY TABLE and DROP ANY TEMPORARY TABLE are not used if ALTER ANY TABLE or DROP ANY TABLE is granted.
- The SHOW PRIVILEGES ON DATABASE statement displays the current active system privileges. This is based on the current user system privileges, system privileges inherited from granted roles, inherited privileges from the database access control list and OpenVMS process privileges.

It does not display any privileges for features which are not enabled for the database. CATALOG and SCHEMA privileges will not be displayed if MULTISCHEMA is not enabled. SYNONYM privileges will not be displayed if SYNONYMS are not enabled. However, the SHOW USER and SHOW ROLE statements will display all granted privileges even if that privilege has no application in the current database configuration.

## 2.1.7 Database Vault Feature

This release of Oracle Rdb introduces the DATABASE VAULT functionality.

The goal of DATABASE VAULT is to avoid accidental database access by an OpenVMS privileged user when the database security policy (ACL) should prevent such access.

This feature allows the database administrator to enforce an access policy for all attached database users by disabling the use of OpenVMS privileges as overrides to the database access control list.

DATABASE VAULT can be enabled by any of these commands.

- The SQL CREATE DATABASE ... DATABASE VAULT IS ENABLED statement.
- The SQL ALTER DATABASE ... DATABASE VAULT IS ENABLED statement.
- The SQL IMPORT DATABASE ... DATABASE VAULT IS ENABLED statement.

---

### Note

---

If a database with DATABASE VAULT enabled is exported, then an IMPORT DATABASE will implicitly execute the DATABASE VAULT IS ENABLED action without that clause being required on the statement.

---

- RMU/SET DATABASE/DATABASE\_VAULT=ENABLED command

DATABASE VAULT can be disabled by any of these commands.

- The SQL ALTER DATABASE ... DATABASE VAULT IS DISABLED statement.
- The SQL IMPORT DATABASE ... DATABASE VAULT IS DISABLED statement.
- RMU/SET DATABASE/DATABASE\_VAULT=DISABLED command

Before executing any of these commands, the user must be granted (at least temporarily) the rights identifier RDBVMS\$DATABASE\_VAULT\_MANAGER that is added to the system during installation.

For example, the SET RIGHTS\_LIST DCL command can be used to temporarily enable it.

```
$ SET RIGHTS_LIST /ENABLE RDBVMS$DATABASE_VAULT_MANAGER
$
$ rmu/set database mf_personnel/database_vault=enable
%RMU-I-MODIFIED, Database state modified
%RMU-W-DOFULLBCK, full database backup should be done to ensure future
recovery
$
$ SET RIGHTS_LIST /DISABLE RDBVMS$DATABASE_VAULT_MANAGER
```

Please refer to the Oracle Rdb SQL Reference Manual, Appendix J for more details. This includes a description of the new DBVAULT audit class that can be used to audit changes to the DATABASE VAULT settings of a database.

## 2.1.8 SET FLAGS Keyword for Hash Join Feature - HASHING

This release of Oracle Rdb has added a flag to control the HASH JOIN feature of the Rdb optimizer. The HASHING flag can be used with the SET FLAGS statement or the RDMS\$SET\_FLAGS logical name to enable this feature. When enabled, the optimizer will attempt to engage the HASH JOIN feature during query solution. The default is NOHASHING(JOINS).

The following example shows the HASHING flag in use.

```
SQL> set flags 'strategy,detail(2)';
SQL> set flags 'hashing(joins)';
SQL>
SQL> select e.employee_id, e.birthday, jh.job_start
cont> from employees e, job_history jh
cont> where e.employee_id = jh.employee_id
cont> and jh.job_end is null
cont> ;
Tables:
  0 = EMPLOYEES
  1 = JOB_HISTORY
Conjunct: 0.EMPLOYEE_ID = 1.EMPLOYEE_ID
Hash Q1
Outer Build
Match_Key:0.EMPLOYEE_ID
Get      Retrieval by index of relation 0:EMPLOYEES
Index name EMP_EMPLOYEE_ID [0:0]
Inner Probe
Match_Key:1.EMPLOYEE_ID
Conjunct: MISSING (1.JOB_END)
Get      Retrieval by index of relation 1:JOB_HISTORY
Index name JH_EMPLOYEE_ID [0:0]
Table=0:EMPLOYEES #Buckets=131 #Hits=61 #Collisions=39 #Dups=0 #Dups_Chain=0
Load_Factor= 4.656488549618321E-001
E.EMPLOYEE_ID E.BIRTHDAY JH.JOB_START
```

```

00164          28-Mar-1947   21-Sep-1981
00165          15-May-1954    8-Mar-1981
.
.
.
100 rows selected
SQL>

```

To disable the flag, use 'NOHASHING(JOINS)'. The setting is displayed by the SHOW FLAGS statement.

See also the new logical name RDMSS\$ENABLE\_HASH\_JOIN. Defining this logical name to true ("T", "t", "Y", "y" or "1") instructs the Rdb optimizer to try to use in-memory HASH JOIN to solve queries.

## 2.1.9 JOIN BY HASH Clause in CREATE OUTLINE Statement

This release of Oracle Rdb adds a new JOIN BY HASH clause to the CREATE OUTLINE statement and the OPTIMIZE OUTLINE clause of the select statement.

The following example shows the new syntax and the resulting query strategy.

```

SQL> create outline QO_1
cont> id '352E2736F133A6A322A3C935DB2CBE12'
cont> mode 0
cont> as (
cont>   query (
cont>     subquery (
cont>       EMPLOYEES 0 access path index EMP_EMPLOYEE_ID
cont>       join by hash to
cont>       SALARY_HISTORY 1 access path index SH_EMPLOYEE_ID
cont>     )
cont>   )
cont> )
cont> compliance optional;
SQL>
SQL> set flags 'strategy,detail(2)';
SQL>
SQL> select
cont>   e.employee_id, sh.salary_start, sh.salary_amount
cont> from
cont>   employees e
cont>   inner join
cont>   salary_history sh on (e.employee_id = sh.employee_id
cont>                        and sh.salary_end is null)
cont> optimize using QO_1
cont> ;
~S: Outline "QO_1" used
Tables:
  0 = EMPLOYEES
  1 = SALARY_HISTORY
Conjunct: 0.EMPLOYEE_ID = 1.EMPLOYEE_ID
Hash Q1
Outer Build
Match Key:0.EMPLOYEE_ID
Index only retrieval of relation 0:EMPLOYEES
Index name EMP_EMPLOYEE_ID [0:0]
Inner Probe
Match Key:1.EMPLOYEE_ID
Conjunct: MISSING (1.SALARY_END)
Get Retrieval by index of relation 1:SALARY_HISTORY
Index name SH_EMPLOYEE_ID [0:0]
Table=0:EMPLOYEES #Buckets=131 #Hits=61 #Collisions=39 #Dups=0 #Dups_Chain=0
Load_Factor= 4.656488549618321E-001

```

E.EMPLOYEE_ID	SH.SALARY_START	SH.SALARY_AMOUNT
00164	14-Jan-1983	\$51,712.00
00165	1-Jul-1982	\$11,676.00

.

.

.

---

**Note**

---

This syntax, JOIN BY HASH, cannot be applied remotely to an older version of Oracle Rdb. An error such as this will be returned.

```
SQL> create outline QO_1
cont> id '352E2736F133A6A322A3C935DB2CBE12'
cont> mode 0
cont> as (
cont>   query (
cont>     subquery (
cont>       EMPLOYEES 0 access path index EMP_EMPLOYEE_ID
cont>       join by hash to
cont>       SALARY_HISTORY 1 access path index SH_EMPLOYEE_ID
cont>     )
cont>   )
cont> )
cont> compliance optional;
%SQL-F-UNSUPVER, Operation is unsupported for version of database
-SQL-F-UNSUPFEATURE, feature JOIN BY HASH is not supported
```

---

## 2.1.10 Hash Join Feature

Status: BETA

This release of Oracle Rdb includes a new optimization method known as Hash Join. A Hash Join is performed by hashing (mapping) one set of data into virtual memory based on the join columns and reading the other table to probe into the hash table to locate matching rows.

Typically, a Hash Join has a lower cost compared to the alternate of sorting when the hash table can be held entirely in memory, with the total cost amounting to very little other than the cost of reading the data sets. The cost rises if the hash table has to be spilled over to a temporary file.

Hash Join is only used for equi-joins. In general, Hash Join is a better solution for joining large numbers of rows in an equi-join.

Applications that in the past used Match Join might (unknowingly) rely on the implicit use of SORT during the query solution. However, as no implicit SORT is performed, the data might appear in a different order with Hash Join. Adding an ORDER BY will sort the result data but no longer sort the inputs to the join.

---

**Note**

---

Not all queries will be solved using HASH JOIN if this optional feature is enabled. Use the SET FLAGS 'STRATEGY,DETAIL(3)' to see a report of the current restrictions which cause the HASH JOIN method to be rejected.

---

## Enabling HASH JOIN

The optimizer, by default, does not try the HASH JOIN method in solutions. This action must be enabled by the programmer in one of the following ways. The optimizer will then include HASH JOIN as part of its solution matching, but it may be rejected for various reasons.

- Define the logical name `RDMS$ENABLE_HASH_JOIN`.  
Defining this logical name to true ("T", "t", "Y", "y" or "1") instructs the Rdb optimizer to try to use in-memory HASH JOIN to solve queries.
- Defining the logical name `RDMS$SET_FLAGS` or using the `SET FLAGS` statement with the string 'HASHING(JOINS)'. See Section 2.1.8 for more details.
- The new `JOIN BY HASH` clause in `CREATE OUTLINE` statement and the `OPTIMIZE OUTLINE` clause of the select statement. See Section 2.1.9 for more details.
- Specifying the `OPTIMIZE FOR HASH JOIN` on the select statement.

```
SQL> select e.employee_id, (sh.salary_end - sh.salary_start) month (3)
cont> from employees e, salary_history sh
cont> where e.employee_id = sh.employee_id
cont>         and e.employee_id <= '00164'
cont>         and sh.salary_end is not null
cont> optimize for hash join
cont> ;
.
.
.
```

## Example

This example shows the strategy used by the optimizer when HASH JOIN is enabled.

```
SQL> set flags 'STRATEGY,DETAIL(2)';
SQL> set flags 'HASHING(JOINS)';
SQL>
SQL> select e.employee_id, (sh.salary_end - sh.salary_start) month (3)
cont> from employees e, salary_history sh
cont> where e.employee_id = sh.employee_id
cont>         and e.employee_id <= '00164'
cont>         and sh.salary_end is not null
cont> ;
Tables:
  0 = EMPLOYEES
  1 = SALARY_HISTORY
Conjunct: 0.EMPLOYEE_ID = 1.EMPLOYEE_ID
Hash Q1
Outer Build
Match Key:0.EMPLOYEE_ID
Index only retrieval of relation 0:EMPLOYEES
  Index name  EMP_EMPLOYEE_ID [0:1]
  Keys: 0.EMPLOYEE_ID <= '00164'
Inner Probe
Match Key:1.EMPLOYEE_ID
Conjunct: NOT MISSING (1.SALARY_END)
Conjunct: 1.EMPLOYEE_ID <= '00164'
Get      Retrieval by index of relation 1:SALARY_HISTORY
  Index name  SH_EMPLOYEE_ID [0:1]
  Keys: 1.EMPLOYEE_ID <= '00164'
Table=0:EMPLOYEES #Buckets=47 #Hits=1 #Collisions=0 #Dups=0 #Dups_Chain=0
```

```

Load_Factor= 2.127659574468085E-002
E.EMPLOYEE_ID
00164          008
00164          006
00164          016
3 rows selected
SQL>

```

**Now try the same query with NOHASHING.**

```

SQL> set flags 'NOHASHING(JOINS)';
SQL>
SQL> select e.employee_id, (sh.salary_end - sh.salary_start) month (3)
cont> from employees e, salary_history sh
cont> where e.employee_id = sh.employee_id
cont>        and e.employee_id <= '00164'
cont>        and sh.salary_end is not null
cont> ;
Tables:
 0 = EMPLOYEES
 1 = SALARY_HISTORY
Conjunct: 0.EMPLOYEE_ID = 1.EMPLOYEE_ID
Match Q1
Outer loop      (zig-zag)
Match_Key:0.EMPLOYEE_ID
Index_Key:EMPLOYEE_ID
  Index only retrieval of relation 0:EMPLOYEES
    Index name  EMP_EMPLOYEE_ID [0:1]
    Keys: 0.EMPLOYEE_ID <= '00164'
Inner loop      (zig-zag)
Match_Key:1.EMPLOYEE_ID
Index_Key:EMPLOYEE_ID
  Conjunct: NOT MISSING (1.SALARY_END)
  Conjunct: 1.EMPLOYEE_ID <= '00164'
  Get      Retrieval by index of relation 1:SALARY_HISTORY
    Index name  SH_EMPLOYEE_ID [0:1]
    Keys: 1.EMPLOYEE_ID <= '00164'
E.EMPLOYEE_ID
00164          008
00164          006
00164          016
3 rows selected
SQL>

```

### 2.1.11 ALTER DATABASE ... LOAD ACL IDENTIFIERS Clause

In this release of Oracle Rdb, the database administrator can automatically and simply create users and roles in the database. This clause, **LOAD ACL IDENTIFIERS**, is part of the **ALTER DATABASE** statement and can be run as often as necessary to add new users and roles derived from the existing access control lists (ACLs) granted to the database and database objects.

The following example shows this clause on a sample database:

```

SQL> attach 'filename WAREHOUSE';
SQL>
SQL> show roles;
Roles in database with filename WAREHOUSE
  No roles found
SQL> show users;
Users in database with filename WAREHOUSE
  No users found
SQL>
SQL> disconnect all;
SQL>
SQL> alter database
cont>     filename WAREHOUSE
cont>
cont>     load acl identifiers
cont> ;
SQL>
SQL> attach 'filename WAREHOUSE';
SQL>
SQL> show roles;
Roles in database with filename WAREHOUSE
  CDD$EXTENDER
  CDD$SYSTEM
  STORES_CUST_READABLE
  STORES_EXTRACT_TEXT
  STORES_MAIL_TEXT
  STORES_PRINT_TEXT
  STORES_USER
SQL> show users;
Users in database with filename WAREHOUSE
  FLEE
  ISMITH
  JJONES
  KSTJOHN
  WH_QUERY_1
  WH_QUERY_2
  WH_QUERY_3
  WH_QUERY_4
  WH_QUERY_5
SQL>
SQL> disconnect all;
SQL>

```

When the database administrator uses the GRANT statement to give access to users and OpenVMS rights identifiers (aka roles), they are recorded in the access control lists for each object; database, table, view, column, sequence, module, and routine. This clause of the ALTER DATABASE statement reads every ACL in the database and creates USER and ROLE definitions if necessary.

---

**Note**

---

Some access control entries (ACEs) may use OpenVMS group identifiers (PUBLIC, [\*,\*], [ADMIN,\*], [\*,], etc), or special modifier rights identifiers (BATCH, DIALUP, INTERACTIVE, LOCAL, NETWORK, REMOTE) which are not valid users and roles - these will be ignored by the LOAD ACL IDENTIFIERS clause.

---

In addition to the DBADM privilege required to use ALTER DATABASE, this clause also requires SECURITY on the database. Alternately, the user must be granted the ALTER ANY DATABASE, CREATE ANY USER and CREATE ANY ROLE database system privilege.

## 2.1.12 ALTER TABLE Actions for READ ONLY Table

This release of Oracle Rdb adds the ability to change a table to READ ONLY access. Once committed, no other application or interactive SQL session may modify rows in the table. You can issue database definition statements (DDL) as long as they do not modify any table data. Operations on indices associated with the table are allowed when the table is in READ ONLY mode. To revert to a read-write table, the clause READ WRITE can be applied.

While the table is READ ONLY the following restrictions apply:

- The data manipulation statements INSERT, UPDATE, DELETE may not modify rows in the table.
- Table updates via LIST cursor may fail during the OPEN or CLOSE statement depending on the cursor declaration.
- The SELECT statement using the FOR UPDATE clause may fail because it tries to apply UPDATE semantics.

```
SQL> select * from SAMPLE_TABLE for update;
%RDB-E-READ_ONLY_REL, relation SAMPLE_TABLE was reserved for read access;
updates not allowed
```

- The TRUNCATE TABLE statement is not permitted to truncate rows from the table.
- ALTER TABLE ... ADD COLUMN is permitted unless a DEFAULT is added for the new column (either explicitly or implicitly from a domain reference). In this case, Rdb would normally execute an UPDATE statement to include the default into each pre-existing row.
- Most other database definition statements are permitted. For instance, CREATE INDEX, ALTER INDEX ... REBUILD ALL PARTITIONS, DROP INDEX, can all be performed while the table is in this state.

The following example shows the diagnostic reported by Oracle Rdb.

```
SQL> alter table SAMPLE_TABLE
cont>   read only
cont> ;
SQL>
SQL> show table (comment) SAMPLE_TABLE;
Information for table SAMPLE_TABLE

Comment on table SAMPLE_TABLE:
Samples table

Table is set READ ONLY

SQL> truncate table SAMPLE_TABLE;
%RDB-E-NO META UPDATE, metadata update failed
-RDB-E-READ_ONLY_REL, relation SAMPLE_TABLE was reserved for read access;
updates not allowed
SQL> commit;
SQL>
```

### 2.1.13 NULLS FIRST and NULLS LAST Options for ORDER BY Clause

This release of Oracle Rdb adds the NULLS FIRST and NULLS LAST options to the ORDER BY clause. These options control the ordering of the NULL values relative to the ordering of the key data. This is demonstrated by the four examples shown below.

```
SQL> select employee_id, salary_start, salary_end, salary_amount
cont> from salary_history sh
cont> where employee_id = '00164'
cont> order by salary_end asc nulls first
cont> ;
```

EMPLOYEE_ID	SALARY_START	SALARY_END	SALARY_AMOUNT
00164	14-Jan-1983	NULL	\$51,712.00
00164	5-Jul-1980	2-Mar-1981	\$26,291.00
00164	2-Mar-1981	21-Sep-1981	\$26,291.00
00164	21-Sep-1981	14-Jan-1983	\$50,000.00

4 rows selected

SQL>

```
SQL> select employee_id, salary_start, salary_end, salary_amount
cont> from salary_history sh
cont> where employee_id = '00164'
cont> order by salary_end asc nulls last
cont> ;
```

EMPLOYEE_ID	SALARY_START	SALARY_END	SALARY_AMOUNT
00164	5-Jul-1980	2-Mar-1981	\$26,291.00
00164	2-Mar-1981	21-Sep-1981	\$26,291.00
00164	21-Sep-1981	14-Jan-1983	\$50,000.00
00164	14-Jan-1983	NULL	\$51,712.00

4 rows selected

SQL>

```
SQL> select employee_id, salary_start, salary_end, salary_amount
cont> from salary_history sh
cont> where employee_id = '00164'
cont> order by salary_end desc nulls first
cont> ;
```

EMPLOYEE_ID	SALARY_START	SALARY_END	SALARY_AMOUNT
00164	14-Jan-1983	NULL	\$51,712.00
00164	21-Sep-1981	14-Jan-1983	\$50,000.00
00164	2-Mar-1981	21-Sep-1981	\$26,291.00
00164	5-Jul-1980	2-Mar-1981	\$26,291.00

4 rows selected

SQL>

```
SQL> select employee_id, salary_start, salary_end, salary_amount
cont> from salary_history sh
cont> where employee_id = '00164'
cont> order by salary_end desc nulls last
cont> ;
```

EMPLOYEE_ID	SALARY_START	SALARY_END	SALARY_AMOUNT
00164	21-Sep-1981	14-Jan-1983	\$50,000.00
00164	2-Mar-1981	21-Sep-1981	\$26,291.00
00164	5-Jul-1980	2-Mar-1981	\$26,291.00
00164	14-Jan-1983	NULL	\$51,712.00

4 rows selected

SQL>

## 2.1.14 Enhancements to RMU Unload After\_Image (LogMiner) Interface

This release of Oracle Rdb adds support for the following new options for the RMU Unload After\_Image (aka LogMiner) command.

- The RMU Unload After\_Image command can now output an XML script containing the updates to the selected table or tables.
- When the FORMAT qualifier specifies one of DELIMITED\_TEXT, DUMP or XML, the TRIM option can also be used to trim leading and/or trailing spaces.
- The SYMBOLS qualifier now accepts the keyword LOCAL (default) or GLOBAL. The default behavior is to generate local scope DCL symbols. When Symbols=GLOBAL is used, these symbols have global scope.

### 2.1.14.1 XML Option to FORMAT Qualifier

When using FORMAT=XML, the following options can also be specified:

- CHARACTER\_ENCODING\_XML

When using RMU Unload After\_Image Format=XML, the XML header record will, by default, use the character encoding "ISO-8859-1". For example, this will appear in the header of the XML file.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

This encoding (ISO-8859-1) is Latin 1 and covers encoding of many European character sets. However, this encoding is not adequate if you use other character encoding for Asian languages, or languages not covered by this ISO Standard.

This release of Oracle Rdb adds an option, CHARACTER\_ENCODING\_XML, that allows the command procedure to specify an alternate character encoding. For example, if you wish to have the XML header describe UTF8, then specify the qualifier /FORMAT=(XML,CHAR="utf-8").

```
<?xml version="1.0" encoding="utf-8"?>
```

- DATA\_XML\_NULL

This option accepts one of the following keywords which control the output of NULL column values: DROP, NIL\_ATTRIBUTE, or EMPTY. If this option is not specified, the default is EMPTY as shown in the following example.

This is a fragment of the XML data generated by RMU Unload After\_Image with the qualifier /FORMAT=(XML,TRIM=TRAILING) defaulting to DATA\_XML\_NULL=EMPTY.

```
<ROW>
  <RDB_LM_ACTION>M</RDB_LM_ACTION>
  <RDB_LM_RELATION_NAME>SAMPLE2</RDB_LM_RELATION_NAME>
  <RDB_LM_RECORD_TYPE>32</RDB_LM_RECORD_TYPE>
  <RDB_LM_DATA_LEN>65254</RDB_LM_DATA_LEN>
  <RDB_LM_NBV_LEN>3</RDB_LM_NBV_LEN>
  <RDB_LM_DBK>60:14417:0</RDB_LM_DBK>
  <RDB_LM_START_TAD>2020-05-11T10:20:50.88</RDB_LM_START_TAD>
  <RDB_LM_COMMIT_TAD>2020-05-11T10:20:50.89</RDB_LM_COMMIT_TAD>
  <RDB_LM_TSN>712</RDB_LM_TSN>
  <RDB_LM_RECORD_VERSION>1</RDB_LM_RECORD_VERSION>
  <IDENT>101</IDENT>
  <COMMENT/>
  <DETAILS/>
</ROW>
```

If DATA\_XML\_NULL is specified as DROP, then that column's value is omitted from the XML record. If DATA\_XML\_NULL is specified as NIL\_ATTRIBUTE, then the XML tag attribute for the column is output as xsi:nil="true".

- **TRIM**

This option allows the data values for the columns to have trailing and leading spaces and horizontal tab characters removed from the columns.

#### 2.1.14.2 TRIM Option

When the FORMAT selected is one of DELIMITED\_TEXT, DUMP or XML, then RMU can be instructed to trim trailing and/or leading spaces and horizontal tab characters from the columns.

- The default when FORMAT is XML or DELIMITED\_TEXT is no trimming. The default when FORMAT is DUMP is TRIM\_TRAILING.
- The TRIM option is not compatible with FORMAT=BINARY and FORMAT=TEXT.
- One of the following keywords can be specified for TRIM: TRAILING, LEADING and BOTH. If TRIM is specified without qualification, then TRAILING is assumed.

The following example shows the use of format XML with the TRIM=BOTH option.

```
$      RMU/UNLOAD-  
      /AFTER_IMAGE -  
      /INCLUDE=ACTION: (COMMIT,DELETE,NOMODIFY) -  
      /LOG-  
      /TABLE=(name=SAMPLE2, output=SAMPLE3.DAT) -  
      /FORMAT=(XML,TRIM=BOTH) -  
      /ORDER_AIJ_FILES-  
      USER1: [TESTER.LOGMINER] LOGMINER_DB -  
      USER1: [TESTER.LOGMINER] AIJ_BU_*.BAIJ  
$
```

#### Usage Notes

- When unloading rows with columns that have many trailing spaces, then using FORMAT=(XML,TRIM) or FORMAT=(DELIMITED\_TEXT,TRIM) can, in some cases, reduce the size of the output file without loss of significant data.
- When dumping rows with long columns that have many trailing spaces, then using /FORMAT=(DUMP,TRIM) can significantly reduce the size of the output file. Therefore, RMU Unload After\_Image implicitly enables TRIM=TRAILING.

#### 2.1.14.3 SYMBOLS Qualifier

The SYMBOLS qualifier now accepts the keyword LOCAL (default) or GLOBAL. The default behavior is to generate local scope DCL symbols. When Symbols=GLOBAL is used, these symbols have global scope.

## Arguments

Symbols

Symbols=LOCAL

Symbols=GLOBAL

NoSymbols

Specifies whether DCL symbols are to be created, indicating information about records extracted for each table.

The default is Symbols, which causes local symbols to be created. Use Symbols=GLOBAL to have RMU define global symbols. Use NoSymbols to prevent creation of any DCL symbols.

If a large number of tables are being unloaded, too many associated symbols may be created and the CLI symbol table space can become exhausted. The error message "LIB-F-INSLIMEM, insufficient CLI memory" is returned in this case. Specify the Nosymbols qualifier to prevent creation of the symbols.

### 2.1.15 Named Partition Support for RESERVING Clause

This release of Oracle Rdb adds the ability to use named partitions in the RESERVING clause of the SET TRANSACTION or DECLARE TRANSACTION statements. In prior versions, only partition numbers were allowed.

The partition names might be system generated (as shown below for the EMPLOYEES\_MAP from the MF\_PERSONNEL database) or they can be defined as part of the CREATE STORAGE MAP statement.

The following example shows the partition numbers as well as the system generated partition names under the *Partition information for storage map* output.

```
SQL> show storage map employees_map
      EMPLOYEES_MAP
For Table:      EMPLOYEES
Placement Via Index:  EMPLOYEES_HASH
Partitioning is:  UPDATABLE

Partition information for storage map:
Compression is:  ENABLED
Partition: (1) SYS_P00079
Storage Area: EMPIDS_LOW
Partition: (2) SYS_P00080
Storage Area: EMPIDS_MID
Partition: (3) SYS_P00081
Storage Area: EMPIDS_OVER

SQL>
```

#### Usage Notes

- The PARTITION clause accepts a list of partition names or a list of partition ordinal values. You may not mix numeric and named notations.
- Duplicate partition names in the RESERVING clause will cause an exception. Review the RESERVING clause and correct the partition names.

```

set transaction
  read write
  evaluating
    job_history_foreign1 at verb time
    ,salary_history_foreign1 at verb time
  reserving
    employees partition (SYS_P00080, SYS_P00080) for exclusive write
;
%RDB-E-BAD_TPB_CONTENT, invalid transaction parameters in the transaction
parameter block (TPB)
-RDMS-E-DUPPARTNAME, partition SYS_P00080 for table EMPLOYEES already used

```

- **Unknown partition names in the RESERVING clause (which might occur due to a change in the storage map definition) will cause an exception. Use the SHOW STORAGE MAP statement to review the partition names.**

```

create module mod_testing1a
  language sql
  procedure proc_xa ();
begin not atomic
set transaction
  read write
  evaluating salary_history_foreign1 at verb time
  reserving employees partition (SYS_P00080, "UNKNOWN", SYS_P00081)
    ,departments for protected write;
  commit;
end;
end module;
%RDB-E-NO_META_UPDATE, metadata update failed
-RDB-E-BAD_TPB_CONTENT, invalid transaction parameters in the transaction
parameter block (TPB)
-RDMS-F-PARTNEXTS, partition "UNKNOWN" does not exist in this map or index
"EMPLOYEES_MAP"

```

## 2.2 Discontinued Feature Support

### 2.2.1 RMU Backup No Longer Supports HUFFMAN or LZSS Compression, Use ZLIB Instead

This release of Oracle Rdb removes the compression options HUFFMAN and LZSS from the RMU Backup and RMU Backup After\_Journal commands.

These older compression algorithms are much slower than the default ZLIB compression. Orders of magnitude more CPU is required in some cases. If you receive a diagnostic as shown in the following example, then Oracle recommends accepting the default, changing the DCL command procedure, or RMU PLAN file to explicitly state ZLIB. Additionally, ZLIB compression allows the database administrator to determine levels of compression efficiency, from less time to more effective compression. Refer to Oracle Rdb RMU Reference Manual for more details.

The following example shows the new diagnostic reported by RMU.

```

$ rmu/backup-
  /disk_file-
  /list_plan=plan_1.plan-
  /execute-
  /compress=lzss-
  /parallel=Executor_Count=3 -
  sql$database -
  sav_l1.rbf,sav_l2.rbf,sav_l3.rbf
%RMU-E-NOTSUPFORVER, The function COMPRESSION.LZSS is not supported for Oracle
Rdb V7.4-100
-RMU-I-COMPUSEZLIB, use the default, or specify ZLIB compression
%RMU-F-FTL_BCK, Fatal error for BACKUP operation at 18-JUN-2020 15:45:23.27
$

```

**Also note that neither HUFFMAN nor LZSS are accepted by the RMU Set After\_Journal Backups qualifier. That command has always required the preferred ZLIB algorithm for compression.**



---

## Optimizer Enhancements

### A.1 Optimizer Enhancements

#### A.1.1 Changes and Improvements to the Rdb Optimizer and Query Compiler

These features fall generally under the title *query rewrite*, and allow the query compiler to present a simplified query for optimization and execution.

- CAST function elimination

In most cases, CAST actions must be executed at runtime to convert from the source data type to that specified by the CAST function. However, in some cases, the Rdb query compiler can eliminate or replace the CAST function with a literal value during query compile. This saves CPU time as the action is performed just once rather than once per row processed.

This replacement includes the following:

- When CAST of DATE (ANSI), DATE (VMS) or TIMESTAMP data types is performed to a compatible type of DATE or TIMESTAMP, then in many cases the CAST operator is not required.
- CAST of string literals to DATE (ANSI), DATE (VMS), TIME, TIMESTAMP and INTERVAL can be processed at compile time. For example, `CAST('2013-1-1' AS DATE ANSI)` is implicitly converted to a DATE literal `DATE'2013-1-1'`.
- CAST of small integer values is now done by the compiler. For example, `CAST(1 AS SMALLINT)` can be performed at compile time.
- CAST of fixed length (CHAR) literal strings to varying length strings (VARCHAR) is now processed by the compiler if the character set is the same and the target VARCHAR is long enough to hold the source string, as seen in the following example:

```
CAST('TABLE' AS VARCHAR(31))
```

- Constant Folding

Simple arithmetic expressions involving integer or floating point literals are evaluated by the query compiler. The overall effect is smaller executable code and some reduced CPU time for queries. FLOAT, REAL, and DOUBLE PRECISION values are combined to produce DOUBLE PRECISION results. Integer literals (with no fractional component) are combined to produce BIGINT results.

The side effect is that some expressions may now return DOUBLE PRECISION or BIGINT results where in prior versions they produced smaller precision results. This should not affect applications which fetch values into different data types as Oracle Rdb will perform an implicit conversion.

This optimization includes the following:

- \* Addition (+)
- \* Subtraction (-)
- \* Multiplication (\*)
- \* Division (/)

Note that division is not performed at compile time if the divisor is a literal zero (0). Operations which are coded to explicitly divide by zero are probably expected to produce an error at runtime. Although using the SQL SIGNAL statement is now preferred, this technique has been used to terminate procedures when an incorrect input is encountered.

- Algebraic Rules

Additive identity (zero) can be added to an expression without changing the value. The query compiler will eliminate the literal zero (0) from the expression.

Multiply by zero will result in zero if the other operand is a not nullable expression. In this case, the expression will be replaced by zero.

Multiplicative identity (one) can be multiplied by an expression without changing the value. The query compiler will eliminate the literal one (1) from the expression.

The side effect is that some expressions may now return slightly different data types because the literal is no longer considered as part of the data type computation.

- Simple Predicate Elimination

When predicates include comparison of simple expressions, then the query compiler will attempt to eliminate them from the query predicate. For example, WHERE ('A' = 'A') will be replaced by TRUE, WHERE (2 <> 2) will be replaced with FALSE, and so on.

- Not Nullable Aware

The query compiler is now aware of which columns have a NOT NULL NOT DEFERRABLE constraint enabled. Additionally, this attribute is also implied from any PRIMARY KEY NOT DEFERRABLE constraints.

Using this knowledge, the query compiler can reduce (prune) the query expression. This list defines the ways in which this can occur:

- \* When IS NULL is applied to a not nullable column or expression, then this predicate is replaced with FALSE.
- \* When IS NOT NULL is applied to a not nullable column or expression, then this predicate is replaced with TRUE.

The side effect is that constraints for a table are now loaded for SELECT statements.

This optimization can be disabled using the SET FLAGS statement, or the RDMS\$SET\_FLAGS logical name with the value NOREWRITE(IS\_NULL). The default is REWRITE(IS\_NULL).

- Replace comparisons with NULL

Queries that erroneously compare value expressions with NULL will now be replaced with a simplified UNKNOWN value. For example, a query that uses WHERE EMPLOYEE\_ID = NULL will never find matching rows, because the results of the comparison (equals, not equals, greater than, less than, and so on) are always UNKNOWN.

This optimization can be disabled using the SET FLAGS statement, or the RDMS\$SET\_FLAGS logical name with the value NOREWRITE(UNKNOWN). The default is REWRITE(UNKNOWN).

- **Predicate Pruning**

The AND, OR and NOT operators can be simplified if the logical expressions have been reduced to TRUE, FALSE or UNKNOWN expressions. Depending on the operation, the Rdb query compiler might be able to eliminate the Boolean operator and part of the expression.

This optimization can be disabled using the SET FLAGS statement, or the RDMS\$SET\_FLAGS logical name with the value NOREWRITE(BOOLEANS). The default is REWRITE(BOOLEANS).

- **CASE Expression Pruning**

The prior transformation will also be applied to the Boolean WHEN expressions of a conditional expression (CASE, DECODE, NULLIF, COALESCE, NVL, NVL2, SIGN, ABS, and so on).

In some cases, the resulting conditional expression might resolve to an equivalent conditional expression with fewer branches (some WHEN ... THEN clauses being eliminated) or a simple expression with no conditional expression (all WHEN ... THEN clauses are eliminated).

- **IN Operator Simplification**

The IN operator using a subquery looks similar to the EXISTS boolean expression but it differs in its handling of NULL values. If the query compiler knows that neither source field nor the value set contains NULL, then the EXISTS expression can replace the IN operator. The EXISTS expression generates a better query solution in almost all cases.

This optimization can be disabled using the SET FLAGS statement, or the RDMS\$SET\_FLAGS logical name with the value NOREWRITE(IN\_CLAUSE). The default is REWRITE(IN\_CLAUSE).

In most cases, the results of these optimizations will be transparent to the application. However, database administrators that use SET FLAGS 'STRATEGY,DETAIL' will notice new notations in the displayed strategy.

The following examples show the types of likely results.

In this example, the logical expression (1 = 2) is replaced with FALSE, the logical expression (1 = 1) is replaced with TRUE and the predicate is reduced to just the IS NULL (aka MISSING) check.

```
SQL> select last_name
cont> from employees
cont> where ((1 = 1) and employee_id is null)
cont>         or
cont>         ((1 = 2) and employee_id = '00164');
Tables:
  0 = EMPLOYEES
Conjunct: MISSING (0.EMPLOYEE_ID)
Get      Retrieval sequentially of relation 0:EMPLOYEES
0 rows selected
```

If there existed a NOT NULL NOT DEFERRABLE constraint on the EMPLOYEE\_ID column, the expression can be further reduced because the NOT NULL constraint means the IS NULL test is always FALSE.

```
SQL> alter table EMPLOYEES
cont>     alter column EMPLOYEE_ID
cont>         constraint NN_EMPLOYEE_ID
cont>             NOT NULL
cont>             NOT DEFERRABLE
cont> ;
SQL>
SQL> select last_name
cont> from employees
cont> where ((1 = 1) and employee_id is null)
cont>        or
cont>        ((1 = 2) and employee_id = '00164');
Tables:
  0 = EMPLOYEES
Conjunct: FALSE
Get      Retrieval sequentially of relation 0:EMPLOYEES
0 rows selected
SQL>
```

### REWRITE Flag

The SET FLAGS statement and the RDMSSSET\_FLAGS logical name can be used to enable or disable some of these rewrite actions. This flag primarily exists for Oracle to test the behavior of the query rewrite changes. It can be used by programmers to revert to pre-V7.3 behavior.

REWRITE enables each rewrite setting and NOREWRITE disables them. Additionally, keywords can be added to REWRITE and NOREWRITE to disable selective rewrite actions.

The following new keywords are added for this release of Oracle Rdb.

- BOOLEANS
- IN\_CLAUSE
- IS\_NULL
- UNKNOWN

## A.1.2 Optimized NOT NULL Constraint Execution

This release of Oracle Rdb introduces a new mechanism to verify NOT NULL constraints which are executed immediately at statement end (that is NOT DEFERRABLE). This new mechanism is more efficient (uses less code and virtual memory) than mechanisms used in prior releases. The cost of the constraint check in these cases is a fixed cost with a very small incremental cost for each extra NOT NULL constraint. The NOT NULL requirement of PRIMARY KEY constraints are also checked in the same way.

In prior releases of Oracle Rdb, each NOT NULL constraint would require its own internal query and each would be evaluated serially against the row just inserted or updated.

The following example shows an INSERT into a simple table with STRATEGY flags enabled. As can be observed, the absence of the strategy display indicates that no optimized query was used to validate these constraints.

```

SQL> set flags 'strategy,detail(2),internal,request_name';
SQL>
SQL> insert into SAMPLE
cont>     default values;
%RDB-E-INTEG_FAIL, violation of constraint SAMPLE_PK caused operation to fail
-RDB-F-ON_DB, on database RDB$DEFAULT_CONNECTION
SQL>
SQL> insert into SAMPLE (iden)
cont>     values (0);
%RDB-E-INTEG_FAIL, violation of constraint SAMPLE_DAT_NOT_NULL caused operation
to fail
-RDB-F-ON_DB, on database RDB$DEFAULT_CONNECTION
SQL>
SQL> insert into SAMPLE
cont>     values (1, 'A');
~Sn: Constraint "SAMPLE_PK" evaluated (verb)
Tables:
  0 = SAMPLE
  1 = SAMPLE
Cross block of 2 entries  Q1
  Cross block entry 1
    Conjunct: 0.DBKEY = <var0>
    Firstn: 1
    Get      Retrieval by DBK of relation 0:SAMPLE
  Cross block entry 2
    Conjunct: <agg0> <> 1
    Aggregate-F2: 0:COUNT-SINGLE (<subselect>) Q2
    Index only retrieval of relation 1:SAMPLE
      Index name  SAMPLE_NDX [1:1]
      Keys: 0.IDEN = 1.IDEN
1 row inserted
SQL>

```

Note that any DEFERRABLE constraints will be executed as in prior versions.

### A.1.3 New BITMAPPED SCAN Clauses Added to OPTIMIZE Clause

This release of Oracle Rdb allows the programmer to specify the clause **OPTIMIZE FOR BITMAPPED SCAN** as part of a query. This clause requests that the query optimizer attempt to use BITMAPPED SCAN if there exists multiple supporting indices in the query. The Rdb query optimizer may ignore this request if only one index is used or if no SORTED RANKED indices would be used to solve the query.

The following example shows the effect of using this new clause.

```

SQL> set flags 'strategy,detail(2)';
SQL>
SQL> select count(*)
cont> from car
cont> where make = 'holden'
cont> and cyear = 1979
cont> and colour = 'blue'
cont> and (ctype = 'sedan' or ctype = 'wagon')
cont> optimize for bitmapped scan
cont> ;
Tables:
  0 = CAR
Aggregate: 0:COUNT (*) Q2
Leaf#01 BgrOnly 0:CAR Card=6047 Bitmapped scan
  Bool: (0.MAKE = 'holden') AND (0.CYEAR = 1979)
        AND (0.COLOUR = 'blue')
        AND ((0.CTYPE = 'sedan') OR (0.CTYPE = 'wagon'))
  BgrNdx1 IYEAR [1:1] Fan=97
    Keys: 0.CYEAR = 1979
  BgrNdx2 ICOLOUR [1:1] Fan=79
    Keys: 0.COLOUR = 'blue'
  BgrNdx3 IMAKE [1:1] Fan=79
    Keys: 0.MAKE = 'holden'
  BgrNdx4 ITYPE [(1:1)2] Fan=79
    Keys: r0: 0.CTYPE = 'wagon'
          r1: 0.CTYPE = 'sedan'
          1
1 row selected
SQL>

```

In previous releases, the programmer would need to define the logical name `RDMS$ENABLE_BITMAPPED_SCAN` as 1, `RDMS$SET_FLAGS` as "BITMAPPED\_SCAN", or use the `SET FLAGS 'BITMAPPED_SCAN'` statement in the application.

#### A.1.4 Query Optimization Improvements for IN Clause

The `EXISTS` and `IN` predicates can often be used interchangeably in queries to check for the existence of values in another result set. If possible, the `EXISTS` query should be the first preference because its structure allows for the best query optimization. However, the semantics of these predicates are not identical when `NULL` values are present in one or both tables, especially when used with the `NOT` operator. Care should be taken to ensure correct query behavior in such cases.

With this release of Oracle Rdb, the optimizer will attempt to transform the `IN` predicate to an `EXISTS` predicate when the source columns are known to be not nullable. Such a transformation will return the same results and additionally present a better query for optimization.

The following example shows the strategy selected for `NOT IN` when the optimization is not (or cannot be) applied.

```

SQL> select s.badge_number
cont> from STAFF s
cont> where s.badge_number NOT IN (select kb.badge_number from KNOWN_BADGES kb)
cont> ;
Tables:
  0 = STAFF
  1 = KNOWN_BADGES
Cross block of 2 entries  Q1
Cross block entry 1
  Index only retrieval of relation 0:STAFF
  Index name  STAFF_I [0:0]
Cross block entry 2
  Conjunct: <agg0> = 0
  Aggregate-F1: 0:COUNT-ANY (<subselect>) Q2
  Conjunct: MISSING (0.BADGE_NUMBER) OR MISSING (1.BADGE_NUMBER) OR (
            0.BADGE_NUMBER = 1.BADGE_NUMBER)
  Index only retrieval of relation 1:KNOWN_BADGES
  Index name  KNOWN_BADGES_I [0:0]
BADGE_NUMBER
         4
1 row selected
SQL>

```

**When the target columns (for example BADGE\_NUMBER) in each table have a NOT DEFERRABLE constraint of the type PRIMARY KEY or NOT NULL, then the following strategy is used. The resulting strategy will likely result in faster query execution.**

```

SQL> select s.badge_number
cont> from STAFF s
cont> where s.badge_number NOT IN (select kb.badge_number from KNOWN_BADGES kb)
cont> ;
Tables:
  0 = STAFF
  1 = KNOWN_BADGES
Conjunct: <agg0> = 0
Match      (Agg Outer Join)  Q1
Outer loop
  Match_Key:0.BADGE_NUMBER
  Index only retrieval of relation 0:STAFF
  Index name  STAFF_I [0:0]
  Inner loop      (zig-zag)
  Match_Key:1.BADGE_NUMBER
  Index_Key:BADGE_NUMBER
  Aggregate-F1: 0:COUNT-ANY (<subselect>) Q2
  Index only retrieval of relation 1:KNOWN_BADGES
  Index name  KNOWN_BADGES_I [0:0]
BADGE_NUMBER
         4
1 row selected
SQL>

```

**This transformation is enabled by default but can be disabled using SET FLAGS 'NOREWRITE(IN\_CLAUSE)' and re-enabled using SET FLAGS 'REWRITE(IN\_CLAUSE)'.**

**This new feature was actually introduced in Oracle Rdb Release 7.3.1 but was inadvertently left out of the release notes.**

## A.1.5 Query Optimization Improvements for DATE ANSI Queries

In prior releases of Oracle Rdb, a query such as the following would not use the index on the source column because the CAST function obscured the column reference from the optimizer.

The following example shows a query that is required to select all the transaction records that appeared on a specific date. That is, the query wants to ignore the time portion during the query.

```
SQL> select posting_timestamp
cont> from TRANSACTION_LOG
cont> where cast (posting_timestamp as DATE ANSI) = date ansi'2015-3-19'
cont> ;
Tables:
  0 = TRANSACTION_LOG
Index only retrieval of relation 0:TRANSACTION_LOG
  Index name  TRANS_NXD [1:1]
  Keys: (0.POSTING_TIMESTAMP >= DATE '2015-03-19') AND (0.POSTING_TIMESTAMP <
        (DATE '2015-03-19' + INTERVAL '1' DAY))
  POSTING_TIMESTAMP
  19-MAR-2015 00:25:21.73
1 row selected
SQL>
```

The Oracle Rdb optimizer now detects that an index column is within the CAST function and rewrites such queries to expose the index column. As can be seen, this query now performs an index range retrieval for all values in the specified date/time range.

## A.1.6 New "Index Counts" Optimization for SORTED Indices

In prior releases of Oracle Rdb, a special optimization was applied to SORTED RANKED indices that reduced the I/O and CPU overhead for counting values within an index. In this release of Oracle Rdb, a similar optimization has been implemented for SORTED indices. The main benefit of this optimization is to greatly reduce the CPU overhead for processing SORTED indices with duplicate values.

The following example shows the new strategy applied for COUNT(\*), COUNT(column), and COUNT(DISTINCT column). Here the column being referenced is the leading segment of a SORTED index.

```
SQL> select count(*) from employees;
Tables:
  0 = EMPLOYEES
Aggregate: 0:COUNT (*) Q2
Index only retrieval of relation 0:EMPLOYEES
  Index name  MI_NDX [0:0]      Index counts
  100
1 row selected
SQL> select count(middle_initial)
cont> from employees where middle_initial = 'A';
Tables:
  0 = EMPLOYEES
Aggregate: 0:COUNT (0.MIDDLE_INITIAL) Q2
Index only retrieval of relation 0:EMPLOYEES
  Index name  MI_NDX [1:1]      Index counts
  Keys: 0.MIDDLE_INITIAL = 'A'
```

```

                                4
1 row selected
SQL> select count(distinct middle_initial)
cont> from employees where middle_initial = 'A';
Tables:
  0 = EMPLOYEES
Aggregate: 0:COUNT (DISTINCT 0:MIDDLE_INITIAL) Q2
Index only retrieval of relation 0:EMPLOYEES
  Index name MI_NDX [1:1]      Index distinct counts
  Keys: 0:MIDDLE_INITIAL = 'A'

                                1
1 row selected
SQL>

```

**This optimization is enabled by default and controlled by the flag COUNT\_SCAN. Use the SET FLAGS 'NOCOUNT\_SCAN' statement to disable this optimization, if necessary.**



## RDO, RDBPRE and RDB\$INTERPRET Features

### B.1 RDO, RDBPRE and RDB\$INTERPRET Features

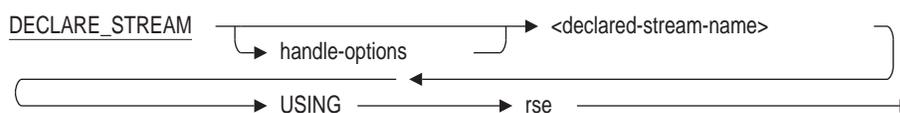
This section describes additions to the RDO and RDBPRE interfaces to Oracle Rdb. Please refer to the Rdb/VMS RDO Reference Manual which contains the latest definition of the RDO language.

#### B.1.1 New Request Options for RDO, RDBPRE and RDB\$INTERPRET

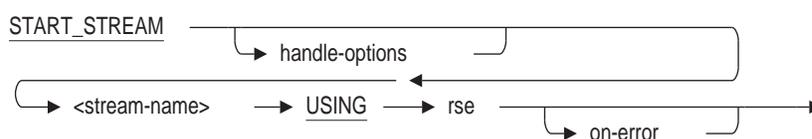
Two new keywords were added to the handle-options for the DECLARE\_STREAM, the START\_STREAM (undeclared format) and FOR loop statements.

The altered statements are shown below.

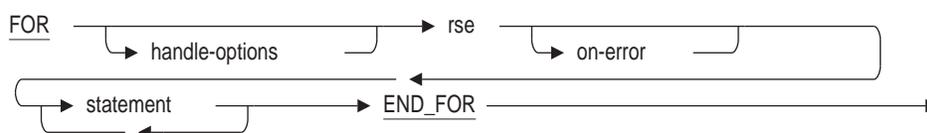
##### DECLARE\_STREAM Format



##### START\_STREAM Format

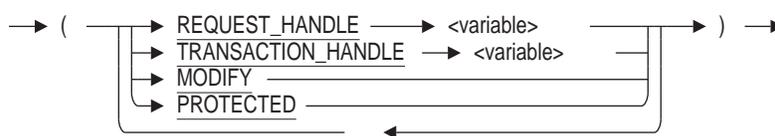


##### FOR Format



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

handle-options =



The following options are available for HANDLE-OPTIONS:

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

This option is valid for RDBPRE, RDB\$INTERPRET, and interactive RDO. This option is not currently available for RDML.

For example:

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

This FOR loop uses the MODIFY option to indicate that the nested MODIFY is an unconditional statement and so aggressive locking can be undertaken during the fetch of the record in the FOR loop.

- **PROTECTED** specifies that the application may modify records fetched by this stream by a separate and independent MODIFY statement. Therefore, this stream should be protected from interference (aka Halloween affect). The optimizer will select a snapshot of the rows and store them in a temporary relation for processing, rather than traversing indexes at the time of the FETCH statement. In some cases, this may result in poorer performance when the temporary relation is large and overflows from virtual memory to a temporary disk file, but the record stream will be protected from interference. The programmer is directed to the documentation for the Oracle Rdb logical names RDM\$BIND\_WORK\_VM and RDM\$BIND\_WORK\_FILE.

This option is valid for RDBPRE, RDB\$INTERPRET, and interactive RDO. This option is not currently available for RDML.

The following example creates a record stream in a BASIC program using Callable RDO:

```
RDMS_STATUS = RDB$INTERPRET ('INVOKE DATABASE PATHNAME "PERSONNEL"')
RDMS_STATUS = RDB$INTERPRET ('START_STREAM (PROTECTED) EMP USING ' + &
                             'E IN EMPLOYEES')

RDMS_STATUS = RDB$INTERPRET ('FETCH EMP')

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

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

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

The problem was corrected in Oracle Rdb Release 7.0.1.

## B.1.2 New Language Features for RDO and Rdb Precompiler

The following new language enhancements have been made to RDO, the Rdb Precompiler (RDBPRE), and the RDO Interpreter (RDB\$INTERPRET).

- LIKE operator

```
--> <value_expr> LIKE <value_expr> ->
```

The rse WITH clause can now specify a LIKE relational operator, which is similar in action to the MATCHING operator. The LIKE operator returns TRUE if the second expression pattern matches the first value expression. LIKE is case sensitive. LIKE uses these special characters:

- % Matches any string
- \_ Matches any character
- \ an escape character. Use \\ to represent a single \, \% to represent a literal "%", and \\_ to represent a literal "\_".

This example is looking for any names starting with one character followed by an apostrophe.

```
RDO> for e in employees
cont>   with e.last_name like ' _'%
cont>   print e.last_name
cont> end_for
LAST_NAME
D'Amico
O'Sullivan
RDO>
```

- FIRST VIA ... FROM sub-query expression

RDO includes a FIRST ... FROM sub-query expression. It returns the value from the matching row. However, if no rows are found, then the query will be aborted with a returned exception.

The following example wishes to list each relation and its associated storage map (if it exists), and shows the reported RDB-E-FROM\_NO\_MATCH error.

```
RDO> for r in rdb$relations
cont>   with r.rdb$system_flag = 0
cont>   sorted by r.rdb$relation_name
cont>   print r.rdb$relation_name,
cont>     first sm.rdb$map_name from sm in rdb$storage_maps with
cont>     sm.rdb$relation_name = r.rdb$relation_name
cont> end_for
RDB$RELATION_NAME          SM.RDB$MAP_NAME
CANDIDATES                 CANDIDATES_MAP
%RDB-E-FROM_NO_MATCH, no record matched the RSE in a "from" expression
RDO>
```

RDO now supports an alternative to the FIRST ... FROM sub-query expression which modifies the behavior when no matching rows were selected by the sub-query. Adding the VIA keyword requests that a MISSING value be returned in such cases, and the query is no longer aborted.

```

RDO> for r in rdb$relations
cont>   with r.rdb$system_flag = 0
cont>   sorted by r.rdb$relation_name
cont>   print r.rdb$relation_name,
cont>     first via sm.rdb$map_name from sm in rdb$storage_maps with
cont>       sm.rdb$relation_name = r.rdb$relation_name
cont> end_for
RDB$RELATION_NAME          SM.RDB$MAP_NAME
CANDIDATES                 CANDIDATES_MAP
COLLEGES                   COLLEGES_MAP
CURRENT_INFO
CURRENT_JOB
CURRENT_SALARY
DEGREES                    DEGREES_MAP
DEPARTMENTS                DEPARTMENTS_MAP
EMPLOYEES                  EMPLOYEES_MAP
EMPS
JOBS                       JOBS_MAP
JOB_HISTORY                JOB_HISTORY_MAP
RESUMES                    RESUMES_MAP
SALARY_HISTORY             SALARY_HISTORY_MAP
WORK_STATUS                WORK_STATUS_MAP
RDO>

```

- **New special functions: RDO\$CURRENT\_USER, RDO\$SESSION\_USER and RDO\$SYSTEM\_USER**

These functions return the user identification of the current, session and system users. They can appear in any place that a field (aka column) can be used. These functions simplify view and trigger definitions created through RDO.

Any view, computed by field, or trigger created by RDO but executed by a SQL session may return different values for each function. However, RDO sessions will typically return the same value from each function.

This query uses the RDO\$CURRENT\_USER function to select the tables and views created by a user.

```

RDO> for r in rdb$relations
cont>   with r.rdb$relation_creator = rdo$current_user
cont>   print r.rdb$relation_name, r.rdb$created
cont> end_for
RDB$RELATION_NAME          RDB$CREATED
EMKP                       23-JUN-2014 12:53:26.28
PICK_HISTORY_REC           31-JUL-2015 15:11:47.46
TEST_TABLE                 8-AUG-2014 08:21:25.96
CUSTOMER_REC               8-AUG-2014 08:17:16.34
TAB1                       8-AUG-2014 08:17:17.88
TAB2                       8-AUG-2014 08:17:17.88
JOB_HIST                   15-AUG-2014 13:49:07.39
SAL_HIST                   15-AUG-2014 13:49:07.39
EMP_NAMES                  14-OCT-2014 21:18:49.03
CAND_NAMES                 14-OCT-2014 21:18:49.03
ACTION_CODES               14-OCT-2014 21:18:49.23
.
.
.

```

### B.1.3 RDO Interface Now Supports Synonym References

This release of Oracle Rdb adds minimal support for synonyms to RDO and RDBPRE. In prior versions, a synonym to a table (or view) was not recognized by the RDO interfaces. For instance, an application built against table names which were subsequently renamed using SQL would no longer compile because the synonyms established by the RENAME TABLE or ALTER TABLE ... RENAME TO statements were not recognized by RDBPRE or RDO.

This support allows queries that reference table or view synonyms to be processed by the RDBPRE precompiler and RDO interactive utility. In addition, most SHOW commands in RDO will recognize a table or view synonym.

Data definition (DDL) commands, such as DROP RELATION or DEFINE CONSTRAINT, do not accept a synonym name as input. For such operations, Oracle recommends using the Interactive SQL interface.

This problem has been corrected in Oracle Rdb Release 7.3.2.0. Synonyms for tables and views created using any of the following statements are now recognized by RDO.

- RENAME TABLE ...
- RENAME VIEW ...
- ALTER TABLE ... RENAME TO ...
- ALTER VIEW ... RENAME TO ...
- CREATE SYNONYMS ... FOR TABLE ...
- CREATE SYNONYMS ... FOR VIEW ...

