17

DBMS_ADVANCED_REWRITE

DBMS_ADVANCED_REWRITE contains interfaces for advanced query rewrite users. Using this package, you can create, drop, and maintain functional equivalence declarations for query rewrite.

See Also:

Oracle Database Data Warehousing Guide for more information about query rewrite

This chapter contains the following topics:

- DBMS_ADVANCED_REWRITE Security Model
- Summary of DBMS_ADVANCED_REWRITE Subprograms

DBMS_ADVANCED_REWRITE Security Model

Default privileges are not granted to anyone for access to DBMS_ADVANCED_REWRITE Security Model procedures. To gain access to these procedures, you must connect as SYSDBA and explicitly grant execute access to the desired database administrators.

You can control security on this package by granting the EXECUTE privilege to selected database administrators or roles. For example, the user er can be given access to use this package by the following statement, executed as SYSDBA:

```
GRANT EXECUTE ON DBMS_ADVANCED_REWRITE TO er;
```

You may want to write a separate cover package on top of this package for restricting the alert names used. Instead of granting the EXECUTE privilege on the DBMS_ADVANCED_REWRITE package directly, you can then grant it to the cover package.

In addition, similar to the privilege required for regular materialized views, the user should be granted the privilege to create an equivalence. For example, the user er can be granted this privilege by executing the following statement as SYSDBA:

```
GRANT CREATE MATERIALIZED VIEW TO er;
```

Summary of DBMS_ADVANCED_REWRITE Subprograms

This table lists the DBMS_ADVANCED_REWRITE subprograms and briefly describes them.

Table 17-1 DBMS_ADVANCED_REWRITE Package Subprograms

Subprogram	Description
ALTER_REWRITE_EQUIVALENCE Procedure	Changes the mode of the rewrite equivalence declaration to the mode you specify

Table 17-1 (Cont.) DBMS_ADVANCED_REWRITE Package Subprograms

Subprogram	Description
BUILD_SAFE_REWRITE_EQUIVALENCE Procedure	Enables the rewrite of top-level materialized views using submaterialized views. Oracle Corporation does not recommend you directly use this procedure
DECLARE_REWRITE_EQUIVALENCE Procedures	Creates a declaration indicating that `source_stmt` is functionally equivalent to `destination_stmt` for as long as the equivalence declaration remains enabled, and that `destination_stmt` is more favorable in terms of performance
DROP_REWRITE_EQUIVALENCE Procedure	Drops the specified rewrite equivalence declaration
VALIDATE_REWRITE_EQUIVALENCE Procedure	Validates the specified rewrite equivalence declaration using the same validation method as described with the `validate` parameter

ALTER_REWRITE_EQUIVALENCE Procedure

This table list the all the package subprograms in alphabetical order.

Syntax

```
DBMS_ADVANCED_REWRITE.ALTER_REWRITE_EQUIVALENCE (
   name              VARCHAR2,
   rewrite_mode      VARCHAR2);
```

Parameters

Table 17-2 ALTER_REWRITE_EQUIVALENCE Procedure Parameters

Parameter	Description
`name`	A name for the equivalence declaration to alter. The name can be of the form `owner.name`, where `owner` complies with the rules for a schema name, and `name` compiles with the rules for a table name. Alternatively, a simple name that complies with the rules for a table name can be specified. In this case, the rewrite equivalence is altered in the current schema. The invoker must have the appropriate alter materialized view privileges to alter an equivalence declaration outside their own schema.

Table 17-2 (Cont.) ALTER_REWRITE_EQUIVALENCE Procedure Parameters

Parameter	Description
`rewrite_mode`	The following modes are supported, in increasing order of power: `disabled`: Query rewrite does not use the equivalence declaration. Use this mode to temporarily disable use of the rewrite equivalence declaration. `text_match`: Query rewrite uses the equivalence declaration only in its text match modes. This mode is useful for simple transformations. `general`: Query rewrite uses the equivalence declaration in all of its transformation modes against the incoming request queries. However, query rewrite makes no attempt to rewrite the specified `destination_query`. `recursive`: Query rewrite uses the equivalence declaration in all of its transformation modes against the incoming request queries. Moreover, query rewrite further attempts to rewrite the specified `destination_query` for further performance enhancements whenever it uses the equivalence declaration. Oracle recommends you use the least powerful mode that is sufficient to solve your performance problem.

BUILD_SAFE_REWRITE_EQUIVALENCE Procedure

This procedure enables the rewrite and refresh of top-level materialized views using submaterialized views. It is provided for the exclusive use by scripts generated by the `DBMS_ADVISOR.TUNE_MVIEW` procedure.

It is required to enable query rewrite and fast refresh when `DBMS_ADVISOR.TUNE_MVIEW` decomposes a materialized view into a top-level materialized view and one or more submaterialized views.

Oracle does not recommend you directly use the `BUILD_SAFE_REWRITE_EQUIVALENCE` procedure. You should use either the `DBMS_ADVISOR.TUNE_MVIEW` or the `DBMS_ADVANCED_REWRITE.CREATE_REWRITE_EQUIVALENCE` procedure as appropriate.

DECLARE_REWRITE_EQUIVALENCE Procedures

This procedure creates a declaration indicating that `source_stmt` is functionally equivalent to `destination_stmt` for as long as the equivalence declaration remains enabled, and that `destination_stmt` is more favorable in terms of performance.

The scope of the declaration is system wide. The query rewrite engine uses such declarations to perform rewrite transformations in `QUERY_REWRITE_INTEGRITY = trusted` and `stale_tolerated` modes.

Because the underlying equivalences between the source and destination statements cannot be enforced by the query rewrite engine, queries can be only rewritten in `trusted` and `stale_tolerated` integrity modes.

Syntax

```
DBMS_ADVANCED_REWRITE.DECLARE_REWRITE_EQUIVALENCE (
   name                   VARCHAR2,
   source_stmt            VARCHAR2,
   destination_stmt       VARCHAR2,
   validate               BOOLEAN    := TRUE,
   rewrite_mode         VARCHAR2   := 'TEXT_MATCH');

DBMS_ADVANCED_REWRITE.DECLARE_REWRITE_EQUIVALENCE (
   name                 VARCHAR2,
   source_stmt          CLOB,
   destination_stmt     CLOB,
   validate             BOOLEAN   := TRUE,
   rewrite_mode         VARCHAR2  := 'TEXT_MATCH');
```

Parameters

Table 17-3   DECLARE_REWRITE_EQUIVALENCE Procedure Parameters

Parameter	Description
`name`	A name for the equivalence declaration. The name can be of the form `owner.name`, where owner complies with the rules for a schema name, and name compiles with the rules for a table name. Alternatively, a simple name that complies with the rules for a table name can be specified. In this case, the rewrite equivalence is created in the current schema. The invoker must have the appropriate `CREATE MATERIALIZED VIEW` privileges to alter an equivalence declaration.
`source_stmt`	A sub-`SELECT` expression in either `VARCHAR2` or `CLOB` format. This is the query statement that is the target of optimization.
`destination_stmt`	A sub-`SELECT` expression in either `VARCHAR2` or `CLOB` format.
`validate`	A Boolean indicating whether to validate that the specified `source_stmt` is functionally equivalent to the specified `destination_stmt`. If `validate` is specified as `TRUE`, `DECLARE_REWRITE_EQUIVALENCE` evaluates the two sub-`SELECT`s and compares their results. If the results are not the same, `DECLARE_REWRITE_EQUIVALENCE` does not create the rewrite equivalence and returns an error condition. If `FALSE`, `DECLARE_REWRITE_EQUIVALENCE` does not validate the equivalence.
`rewrite_mode`	The following modes are supported, in increasing order of power: • `disabled`: Query rewrite does not use the equivalence declaration. Use this mode to temporarily disable use of the rewrite equivalence declaration. • `text_match`: Query rewrite uses the equivalence declaration only in its text match modes. This mode is useful for simple transformations. • `general`: Query rewrite uses the equivalence declaration in all of its transformation modes against the incoming request queries. However, query rewrite makes no attempt to rewrite the specified `destination_query`. • `recursive`: Query rewrite uses the equivalence declaration in all of its transformation modes against the incoming request queries. Moreover, query rewrite further attempts to rewrite the specified `destination_query` for further performance enhancements whenever it uses the equivalence declaration. Oracle recommends you use the least powerful mode that is sufficient to solve your performance problem.

Exceptions

Table 17-4 DECLARE_REWRITE_EQUIVALENCE Procedure Exceptions

Exception	Description
`ORA-30388`	Name of the rewrite equivalence is not specified
`ORA-30391`	The specified rewrite equivalence does not exist
`ORA-30392`	The checksum analysis for the rewrite equivalence failed
`ORA-30393`	A query block in the statement did not write
`ORA-30396`	Rewrite equivalence procedures require the `COMPATIBLE` parameter to be set to 10.1 or greater

Usage Notes

Query rewrite using equivalence declarations occurs simultaneously and in concert with query rewrite using materialized views. The same query rewrite engine is used for both. The query rewrite engine uses the same rewrite rules to rewrite queries using both equivalence declarations and materialized views. Because the rewrite equivalence represents a specific rewrite crafted by a sophisticated user, the query rewrite engine gives priority to rewrite equivalences over materialized views when it is possible to perform a rewrite with either a materialized view or a rewrite equivalence. For this same reason, the cost-based optimizer (specifically, cost-based rewrite) will not choose an unrewritten query plan over a query plan that is rewritten to use a rewrite equivalence even if the cost of the un-rewritten plan appears more favorable. Query rewrite matches properties of the incoming request query against the equivalence declaration's `source_stmt` or the materialized view's defining statement, respectively, and derives an equivalent relational expression in terms of the equivalence declaration's `destination_stmt` or the materialized view's container table, respectively.

DROP_REWRITE_EQUIVALENCE Procedure

This procedure drops the specified rewrite equivalence declaration.

Syntax

```
DBMS_ADVANCED_REWRITE.DROP_REWRITE_EQUIVALENCE (
   name         VARCHAR2);
```

Parameters

Table 17-5 DROP_REWRITE_EQUIVALENCE Procedure Parameters

Parameter	Description
`name`	A name for the equivalence declaration to drop. The name can be of the form `owner.name`, where owner complies with the rules for a schema name, and name compiles with the rules for a table name. Alternatively, a simple name that complies with the rules for a table name can be specified. In this case, the rewrite equivalence is dropped in the current schema. The invoker must have the appropriate drop materialized view privilege to drop an equivalence declaration outside their own schema.

VALIDATE_REWRITE_EQUIVALENCE Procedure

This procedure validates the specified rewrite equivalence declaration.

It uses the same validation method as described with the `VALIDATE` parameter in "VALIDATE_REWRITE_EQUIVALENCE Procedure".

Syntax

```
DBMS_ADVANCED_REWRITE.VALIDATE_REWRITE_EQUIVALENCE (
   name          VARCHAR2);
```

Parameters

Table 17-6 VALIDATE_REWRITE_EQUIVALENCE Procedure Parameters

Parameter	Description
`name`	A name for the equivalence declaration to validate. The name can be of the form owner.name, where owner complies with the rules for a schema name, and name compiles with the rules for a table name. Alternatively, a simple name that complies with the rules for a table name can be specified. In this case, the rewrite equivalence is validated in the current schema. The invoker must have sufficient privileges to execute both the `source_stmt` and `destination_stmt` of the specified equivalence declaration.