171
DBMS_RULE

The `DBMS_RULE` package contains subprograms that enable the evaluation of a rule set for a specified event.

This chapter contains the following topics:

- Overview
- Security Model
- Summary of DBMS_RULE Subprograms

DBMS_RULE Overview

This package contains subprograms that enable the evaluation of a rule set for a specified event.

> **See Also:**
>
> - Rule TYPEs for more information about the types used with the `DBMS_RULE` package
> - DBMS_RULE_ADM

DBMS_RULE Security Model

`PUBLIC` is granted `EXECUTE` privilege on this package.

> **See Also:**
>
> *Oracle Database Security Guide* for more information about user group `PUBLIC`

Summary of DBMS_RULE Subprograms

This table lists the `DBMS_RULE` subprograms and briefly describes them.

Table 171-1 DBMS_RULE Package Subprograms

Subprogram	Description
CLOSE_ITERATOR Procedure	Closes an open iterator
EVALUATE Procedure	Evaluates the rules in the specified rule set that use the evaluation context specified

Table 171-1 (Cont.) DBMS_RULE Package Subprograms

Subprogram	Description
EVALUATE_EXPRESSION Procedure	Evaluates an expression under the logged in user in a session
EVALUATE_EXPRESSION_ITERATOR Procedure	Finds the relevant datapoints and pass `re$value_list` into evaluation interface
EVALUATE_RULE Procedure	Evaluates the condition defined in the Rule
EVALUATE_RULE_ITERATOR Procedure	Finds the relevant datapoints and pass `re$value_list` into evaluation interface
GET_NEXT_HIT Function	Returns the next rule that evaluated to `TRUE` from a true rules iterator, or returns the next rule that evaluated to `MAYBE` from a maybe rules iterator; returns `NULL` if there are no more rules that evaluated to `TRUE` or `MAYBE`.
IS_FAST Procedure	Returns `TRUE` if the expression can be evaluated fast. An expression can be evaluated fast if the engine does not need to run any internal SQL and does not need to go to PL/SQL layer in case there are any PL/SQL functions referred.
GET_NEXT_RESULT Function	Iterates over result from the expression given `result_val_iterator`.

CLOSE_ITERATOR Procedure

This procedure closes an open iterator.

Syntax

```
DBMS_RULE.CLOSE_ITERATOR(
   iterator  IN  BINARY_INTEGER);
```

Parameter

Table 171-2 CLOSE_ITERATOR Procedure Parameter

Parameter	Description
`iterator`	Iterator to be closed

Usage Notes

This procedure requires an open iterator that was returned by an earlier call to `DBMS_RULE.EVALUATE` in the same session. The user who runs this procedure does not require any privileges on the rule set being evaluated.

Closing an iterator frees resources, such as memory, associated with the iterator. Therefore, Oracle recommends that you close an iterator when it is no longer needed.

> **See Also:**
>
> EVALUATE Procedure

EVALUATE Procedure

This procedure evaluates the rules in the specified rule set that use the evaluation context specified for a specified event.

This procedure is overloaded. The `true_rules` and `maybe_rules` parameters are mutually exclusive with the `true_rules_iterator` and `maybe_rules_iterator` parameters. In addition, the procedure with the `true_rules` and `maybe_rules` parameters includes the `stop_on_first_hit` parameter, but the other procedure does not.

Syntax

```
DBMS_RULE.EVALUATE(
   rule_set_name          IN     VARCHAR2,
   evaluation_context     IN     VARCHAR2,
   event_context          IN     SYS.RE$NV_LIST               DEFAULT NULL,
   table_values           IN     SYS.RE$TABLE_VALUE_LIST      DEFAULT NULL,
   column_values          IN     SYS.RE$COLUMN_VALUE_LIST     DEFAULT NULL,
   variable_values        IN     SYS.RE$VARIABLE_VALUE_LIST   DEFAULT NULL,
   attribute_values       IN     SYS.RE$ATTRIBUTE_VALUE_LIST  DEFAULT NULL,

  skip_rules           IN   SYS.RE$RULE_NAME_LIST        DEFAULT NULL,

  dop                  IN   NUMBER,

   result_cache           IN     BOOLEAN                      DEFAULT FALSE,
   stop_on_first_hit      IN     BOOLEAN                      DEFAULT FALSE,
   simple_rules_only      IN     BOOLEAN                      DEFAULT FALSE,
   true_rules             OUT    SYS.RE$RULE_HIT_LIST,
   maybe_rules            OUT    SYS.RE$RULE_HIT_LIST);

DBMS_RULE.EVALUATE(
   rule_set_name          IN     VARCHAR2,
   evaluation_context     IN     VARCHAR2,
   event_context          IN     SYS.RE$NV_LIST               DEFAULT NULL,
   table_values           IN     SYS.RE$TABLE_VALUE_LIST      DEFAULT NULL,
   column_values          IN     SYS.RE$COLUMN_VALUE_LIST     DEFAULT NULL,
   variable_values        IN     SYS.RE$VARIABLE_VALUE_LIST   DEFAULT NULL,
   attribute_values       IN     SYS.RE$ATTRIBUTE_VALUE_LIST  DEFAULT NULL,

  skip_rules            IN   SYS.RE$RULE_NAME_LIST        DEFAULT NULL,

  dop                   IN   NUMBER,

   simple_rules_only      IN     BOOLEAN                      DEFAULT FALSE,
   true_rules_iterator    OUT    BINARY_INTEGER,
   maybe_rules_iterator   OUT    BINARY_INTEGER);
```

Parameters

Table 171-3 EVALUATE Procedure Parameters

Parameter	Description
`rule_set_name`	Name of the rule set in the form `[`*`schema_name`*`.]`*`rule_set_name`*. For example, to evaluate all of the rules in a rule set named `hr_rules` in the `hr` schema, enter `hr.hr_rules` for this parameter. If the schema is not specified, then the schema of the current user is used.

Table 171-3 (Cont.) EVALUATE Procedure Parameters

Parameter	Description
evaluation_context	An evaluation context name in the form [*schema_name.*]*evaluation_context_name*. If the schema is not specified, then the name of the current user is used. Only rules that use the specified evaluation context are evaluated.
event_context	A list of name-value pairs that identify events that cause evaluation
table_values	Contains the data for table rows using the table aliases specified when the evaluation context was created. Each table alias in the list must be unique.
column_values	Contains the partial data for table rows. It must not contain column values for tables, whose values are already specified in table_values.
variable_values	A list containing the data for variables. The only way for an explicit variable value to be known is to specify its value in this list. If an implicit variable value is not specified in the list, then the function used to obtain the value of the implicit variable is invoked. If an implicit variable value is specified in the list, then this value is used and the function is not invoked.
attribute_values	Contains the partial data for variables. It must not contain attribute values for variables whose values are already specified in variable_values.
stop_on_first_hit	If TRUE, then the rules engine stops evaluation as soon as it finds a TRUE rule. If TRUE and there are no TRUE rules, then the rules engine stops evaluation as soon as it finds a rule that may evaluate to TRUE given more data. If FALSE, then the rules engine continues to evaluate rules even after it finds a TRUE rule.
simple_rules_only	If TRUE, then only those rules that are simple enough to be evaluated fast (without issuing SQL) are considered for evaluation. If FALSE, then evaluates all rules.
true_rules	Receives the output of the EVALUATE procedure into a varray of RE$RULE_HIT_LIST type. If no rules evaluate to TRUE, then true_rules is empty. If at least one rule evaluates to TRUE and stop_on_first_hit is TRUE, then true_rules contains one rule that evaluates to TRUE. If stop_on_first_hit is FALSE, then true_rules contains all rules that evaluate to TRUE.
maybe_rules	If all rules can be evaluated completely, without requiring any additional data, then maybe_rules is empty. If stop_on_first_hit is TRUE, then if there is at least one rule that may evaluate to TRUE given more data, and no rules evaluate to TRUE, then maybe_rules contains one rule that may evaluate to TRUE. If stop_on_first_hit is FALSE, then maybe_rules contains all rules that may evaluate to TRUE given more data.
true_rules_iterator	Contains the iterator for accessing rules that are TRUE

Table 171-3 (Cont.) EVALUATE Procedure Parameters

Parameter	Description
maybe_rules_iterator	Contains the iterator for accessing rules that may be TRUE given additional data or the ability to issue SQL
skip_rules	List of rules to skip within this evaluation.
dop	Degree of parallelism
result_cache	If TRUE, Result Cache will be created. If evalate procedure is called with either true_rules_iterator or maybe_rules_iterator, then result_cache is not enabled.

Usage Notes

> **Note:**
>
> Rules in the rule set that use an evaluation context different from the one specified are not considered for evaluation.

The rules in the rule set are evaluated using the data specified for table_values, column_values, variable_values, and attribute_values. These values must refer to tables and variables in the specified evaluation context. Otherwise, an error is raised.

The caller may specify, using stop_on_first_hit, if evaluation must stop as soon as the first TRUE rule or the first MAYBE rule (if there are no TRUE rules) is found.

The caller may also specify, using simple_rules_only, if only rules that are simple enough to be evaluated fast (which means without SQL) should be considered for evaluation. This makes evaluation faster, but causes rules that cannot be evaluated without SQL to be returned as MAYBE rules.

Partial evaluation is supported. The EVALUATE procedure can be called with data for only some of the tables, columns, variables, or attributes. In such a case, rules that cannot be evaluated because of a lack of data are returned as MAYBE rules, unless they can be determined to be TRUE or FALSE based on the values of one or more simple expressions within the rule. For example, given a value of 1 for attribute "a.b" of variable "x", a rule with the following rule condition can be returned as TRUE, without a value for table "tab":

```
(:x.a.b = 1) or (tab.c > 10)
```

The results of an evaluation are the following:

- TRUE rules, which is the list of rules that evaluate to TRUE based on the given data. These rules are returned either in the OUT parameter true_rules, which returns all of the rules that evaluate to TRUE, or in the OUT parameter true_rules_iterator, which returns each rule that evaluates to TRUE one at a time.
- MAYBE rules, which is the list of rules that could not be evaluated for one of the following reasons:
  - The rule refers to data that was unavailable. For example, a variable attribute "x.a.b" is specified, but no value is specified for the variable "x", the attribute "a", or the attribute "a.b".

  - The rule is not simple enough to be evaluated fast (without SQL) and `simple_rules_only` is specified as `TRUE`, or partial data is available.

  Maybe rules are returned either in the `OUT` parameter `maybe_rules`, which returns all of the rules that evaluate to `MAYBE`, or in the `OUT` parameter `maybe_rules_iterator`, which returns each rule that evaluates to `MAYBE` one at a time.

The caller may specify whether the procedure returns all of the rules that evaluate to `TRUE` and `MAYBE` for the event or an iterator for rules that evaluate to `TRUE` and `MAYBE`. A true rules iterator enables the client to fetch each rule that evaluates to `TRUE` one at a time, and a maybe rules iterator enables the client to fetch each rule that evaluates to `MAYBE` one at a time.

If you use an iterator, then you use the `GET_NEXT_HIT` function in the `DBMS_RULE` package to retrieve the next rule that evaluates to `TRUE` or `MAYBE` from an iterator. Oracle recommends that you close an iterator if it is no longer needed to free resources, such as memory, used by the iterator. An iterator can be closed in the following ways:

- The `CLOSE_ITERATOR` procedure in the `DBMS_RULE` package is run with the iterator specified.
- The iterator returns `NULL` because no more rules evaluate to `TRUE` or `MAYBE`.
- The session in which the iterator is running ends.

To run the `DBMS_RULE.EVALUATE` procedure, a user must meet at least one of the following requirements:

- Have `EXECUTE_ON_RULE_SET` privilege on the rule set
- Have `EXECUTE_ANY_RULE_SET` system privilege
- Be the rule set owner

> **Note:**
>
> The rules engine does not invoke any actions. An action context can be returned with each returned rule, but the client of the rules engine must invoke any necessary actions.

> **See Also:**
>
> - Rule TYPEs for more information about the types used with the `DBMS_RULE` package
> - GET_NEXT_HIT Function
> - CLOSE_ITERATOR Procedure

EVALUATE_EXPRESSION Procedure

This procedure allows user to evaluate an expression under the logged in user in a session.

Any re-execute of the same expression with same table alias and variable type will result in reusing the same compiled context. With fixed compile cache size, its possible of aging....

Syntax

```
DBMS_RULE.EVALUATE_EXPRESSION(
   rule_expression          IN         VARCHAR2,
   table_aliases            IN         SYS.RE$TABLE_ALIAS_LIST:= NULL,
   variable_types           IN         SYS.RE$VARIABLE_TYPE_LIST:= NULL,
   table_values             IN         SYS.RE$TABLE_VALUE_LIST:= NULL,
   column_values            IN         SYS.RE$COLUMN_VALUE_LIST:=NULL,
   variable_values          IN         SYS.RE$VARIABLE_VALUE_LIST:=NULL,
   attribute_values         IN         SYS.RE$ATTRIBUTE_VALUE_LIST:=NULL,
   cache                    IN         BOOLEAN DEFAULT FALSE,
   result_val               OUT        BOOLEAN);
```

Parameters

Table 171-4 EVALUATE_EXPRESSION Procedure Parameters

Parameter	Description
rule_expression	Contains an expression string.
table_alias	Contains alias of tables referred in the expression string.
variable_types	Contains type definitions of variables used in expression.
table_values	Contains ROWID of table row for expression evaluation.
column_values	Contains values of columns referred in the expression.
variable_values	Contains values of variables referred in the expression.
attribute_values	Contains values of attributes referred in the expression.
cache	If TRUE, Result Cache will be created.
result_val	Result of the evaluation.

EVALUATE_EXPRESSION_ITERATOR Procedure

This is an user visible interface. Because PL/SQL based callbacks can be expensive, we provide an array based approach. The client program is assumed to find the relevant datapoints and pass re$value_list into evaluation interface. The expression evaluation engine is expected to walk through this list and evaluate expression for each datapoint (re$value_list) element.

Syntax

```
DBMS_RULE.EVALUATE_EXPRESSION_ITERATOR(
       rule_expression          IN             varchar2,
       table_aliases            IN             sys.re$table_alias_list:= NULL,
       variable_types           IN             sys.re$variable_type_list:= NULL,
       values                   IN             sys.re$value_list,
       cache                    IN             boolean DEFAULT FALSE,
       result_val_iter_id       OUT            BINARY_INTEGER)
```

Parameters

Table 171-5 EVALUATE_EXPRESSION_ITERATOR Procedure Parameter

Parameter	Description
rule_expression	Contains an expression string.
table_alias	Alias of tables referred in the above expression string.
variable_types	Type definitions of variables used in expression.
values	List of datapoint values for evaluation.
cache	If TRUE, Result Cache will be created.
result_val_iter_id	Contains iterator for result of array of values sent using value.

EVALUATE_RULE Procedure

The Rule Evaluation API expects that CREATE_RULE procedure has been called with an legitimate EVALUATION_CONTEXT prior. This API will evaluate the condition defined in the Rule.

Syntax

```
DBMS_RULE.EVALUATE_RULE(
  rule_name            IN   VARCHAR2,
  event_context        IN   SYS.RE$NV_LIST               DEFAULT NULL,
  table_values         IN   SYS.RE$TABLE_VALUE_LIST      DEFAULT NULL,
  column_values        IN   SYS.RE$COLUMN_VALUE_LIST     DEFAULT NULL,
  variable_values      IN   SYS.RE$VARIABLE_VALUE_LIST   DEFAULT NULL,
  attribute_values     IN   SYS.RE$ATTRIBUTE_VALUE_LIST  DEFAULT NULL,
  cache                IN   BOOLEAN DEFAULT FALSE,
  result_val           OUT  BOOLEAN);
```

Parameters

Table 171-6 EVALUATE_RULE Procedure Parameter

Parameter	Description
rule_name	Name of the rule previously create using CREATE_RULE procedure.
event_context	A list of name-value pairs that identify events that cause evaluation.
table_values	ROWID of table row for expression evaluation.
column_values	Values of columns referred in the expression
variable_values	Values of variables referred in expression
attribute_values	Values of attributes referred in expression
cache	If TRUE, Result Cache will be created.
result_val	Result of the evaluation

EVALUATE_RULE_ITERATOR Procedure

This is an iterative interface. The client program is assumed to find the relevant datapoints and pass `re$value_list` into evaluation interface.

Evaluation engine is expected to walk through this list and evaluate expression for each datapoint (`re$value_list`) element. User can use `DBMS_RULE.GET_NEXT_RESULT` procedure to iterate through the result list.

Syntax

```
DBMS_RULE.EVALUATE_RULE_ITERATOR)
  rule_name             IN       VARCHAR2,
  event_context         IN       SYS.RE$NV_LIST DEFAULT NULL,
  values                IN       SYS.RE$VALUE_LIST,
  cache                 IN       BOOLEAN DEFAULT FALSE,
  result_val_iter_id    OUT      BINARY_INTEGER);
```

Parameters

Table 171-7 EVALUATE_RULE_ITERATOR Procedure Parameter

Parameter	Description
`rule_name`	Name of the rule previously create using `CREATE_RULE` procedure.
`event_context`	A list of name-value pairs that identify events that cause evaluation
`values`	List of datapoint values for evaluation.
`cache`	If `TRUE`, Result Cache will be created.
`result_val_iter_id`	Contains iterator for result of array of values sent using `values`

GET_NEXT_HIT Function

This function returns the next rule that evaluated to `TRUE` from a true rules iterator, or returns the next rule that evaluated to `MAYBE` from a maybe rules iterator. The function returns `NULL` if there are no more rules that evaluated to `TRUE` or `MAYBE`.

Syntax

```
DBMS_RULE.GET_NEXT_HIT(
   iterator  IN  BINARY_INTEGER)
 RETURN SYS.RE$RULE_HIT;
```

Parameter

Table 171-8 GET_NEXT_HIT Function Parameter

Parameter	Description
`iterator`	The iterator from which the rule that evaluated to `TRUE` or `MAYBE` is retrieved

Usage Notes

This procedure requires an open iterator that was returned by an earlier call to `DBMS_RULE.EVALUATE` in the same session. The user who runs this procedure does not require any privileges on the rule set being evaluated.

When an iterator returns `NULL`, it is closed automatically. If an open iterator is no longer needed, then use the `CLOSE_ITERATOR` procedure in the `DBMS_RULE` package to close it.

> **Note:**
>
> This function raises an error if the rule set being evaluated was modified after the call to the `DBMS_RULE.EVALUATE` procedure that returned the iterator. Modifications to a rule set include added rules to the rule set, changing existing rules in the rule set, dropping rules from the rule set, and dropping the rule set.

> **See Also:**
>
> - Rule TYPEs for more information about the types used with the `DBMS_RULE` package
> - EVALUATE Procedure
> - CLOSE_ITERATOR Procedure

GET_NEXT_RESULT Function

This function iterates over result from the expression given in `RESULT_VAL_ITERATOR`. It returns the expression at iterator evaluated to `TRUE` or `FALSE`.

Syntax

```
DBMS_RULE.GET_NEXT_RESULT (
 result_val_iterator_id    IN   BINARY_INTEGER)
 RETURN  BOOLEAN;
```

Parameter

Table 171-9 GET_NEXT_RESULT Function Parameter

Parameter	Description
`result_val_iterator_id`	Iterator returned from `EVALUATE_EXPRESSION_ITERATOR`

IS_FAST Procedure

Given an expression, of either rule or Independent Expression, this procedure will return `TRUE` if the expression can be evaluated as fast. An expression can be evaluated as fast if the engine

does not need to run any internal SQL and does not need to go to PL/SQL layer in case there are any PL/SQL functions referred.

Syntax

```
DBMS_RULE.IS_FAST(
   expression            IN       VARCHAR2,
   table_aliases         IN       SYS.RE$TABLE_ALIAS_LIST:= NULL,
   variable_types        IN       SYS.RE$VARIABLE_TYPE_LIST:= NULL,
   result_val            OUT      BOOLEAN);
```

Parameter

Table 171-10 IS_FAST Procedure Parameter

Parameter	Description
expression	Expression to check
table_aliases	Alias of tables referred in the above expression string
variable_types	Type definitions of variables used in expression
result_val	If the expression can be evaluated as fast