86
DBMS_FGA

The `DBMS_FGA` package provides fine-grained security functions.

This chapter contains the following topics:

- Security Model
- Operational Notes
- Summary of DBMS_FGA Subprograms

DBMS_FGA Security Model

You must have the `AUDIT_ADMIN` role or the `EXECUTE` privilege on the `DBMS_FGA` package to create audit policies. `DBMS_FGA` is an invoker rights package.

> **Note:**
>
> Starting in Oracle Database 23ai, fine-grained audit policies that are created with the `DBMS_FGA` package will generate audit records in the unified audit trail, viewable with the `UNIFIED_AUDIT_TRAIL` data dictionary view.

To analyze and audit data, you must have the `AUDIT_VIEWER` role. Because the audit function can potentially capture all user environment and application context values, policy administration should be executable by privileged users only. The policy event handler module is executed with the module owner's privilege.

DBMS_FGA Operational Notes

This package is available for only cost-based optimization. The rule-based optimizer may generate unnecessary audit records since audit monitoring can occur before row filtering.

For both the rule-based optimizer and the cost-based optimizer, you can query the `SQL_TEXT` and `SQL_BINDS` columns of the `UNIFIED_AUDIT_TRAIL` view to analyze the SQL text and corresponding bind variables that are issued.

Summary of DBMS_FGA Subprograms

This table describes the `DBMS_FGA` subprograms.

Table 86-1 ***DBMS_FGA Package Subprograms***

Subprogram	Description
ADD_POLICY Procedure	Creates an audit policy using the supplied predicate as the audit condition

Table 86-1 (Cont.) *DBMS_FGA Package Subprograms*

Subprogram	Description
DISABLE_POLICY Procedure	Disables an audit policy
DROP_POLICY Procedure	Drops an audit policy
ENABLE_POLICY Procedure	Enables an audit policy

ADD_POLICY Procedure

This procedure creates an audit policy using the supplied predicate as the audit condition.

Syntax

```
DBMS_FGA.ADD_POLICY(
   object_schema      IN  VARCHAR2 DEFAULT NULL,
   object_name        IN  VARCHAR2,
   policy_name        IN  VARCHAR2,
   audit_condition    IN  VARCHAR2 DEFAULT NULL,
   audit_column       IN  VARCHAR2 DEFAULT NULL,
   handler_schema     IN  VARCHAR2 DEFAULT NULL,
   handler_module     IN  VARCHAR2 DEFAULT NULL,
   enable             IN  BOOLEAN DEFAULT TRUE,
   statement_types    IN  VARCHAR2 DEFAULT SELECT,
   audit_trail        IN  BINARY_INTEGER DEFAULT NULL,
   audit_column_opts  IN  BINARY_INTEGER DEFAULT ANY_COLUMNS,
   policy_owner       IN  VARCHAR2 DEFAULT NULL);
```

Parameters

Table 86-2 ADD_POLICY Procedure Parameters

Parameter	Description
object_schema	Schema of the object to be audited. If NULL, the current schema is assumed.
object_name	Name of the object to be audited
policy_name	Unique name of the policy. Do not enter special characters such as spaces or commas. If you want to use special characters for the policy name, then enclose the name in quotation marks.
audit_condition	A condition in a row that indicates a monitoring condition. NULL is allowed and acts as TRUE.
audit_column	Columns to be checked for access. These can include OLS hidden columns or object type columns. The default, NULL, causes audit if any column is accessed or affected.
handler_schema	Schema that contains the event handler. The default, NULL, causes the current schema to be used.
handler_module	Function name of the event handler; includes the package name if necessary. This function is invoked only after the first row that matches the audit condition in the query is processed. If the procedure fails with an exception, the user SQL statement will fail as well.
enable	Enables the policy if TRUE, which is the default

Table 86-2 (Cont.) ADD_POLICY Procedure Parameters

Parameter	Description
statement_types	SQL statement types to which this policy is applicable: INSERT, UPDATE, DELETE, or SELECT only
audit_trail	Do not set this parameter; it is desupported. All audit records are written to the unified audit trail, viewable by querying the UNIFIED_AUDIT_TRAIL data dictionary view.
audit_column_opts	Establishes whether a statement is audited when the query references *any* column specified in the audit_column parameter or only when *all* such columns are referenced
policy_owner	User who owns the fine-grained auditing policy. However, this setting is not a user-supplied argument. The Oracle Data Pump client uses this setting internally to recreate the fine-grained audit policies appropriately.

Usage Notes

- A table or view can have a maximum of 256 fine-grained audit policies applied to it.
- If object_schema is not specified, the current schema is assumed.
- An FGA policy should not be applied to out-of-line columns such as LOB columns.
- Each audit policy is applied to the query individually. However, at most one audit record may be generated for each policy, no matter how many rows being returned satisfy that policy's audit_condition. In other words, whenever any number of rows being returned satisfy an audit condition defined on the table, a single audit record will be generated for each such policy.
- If a table with an FGA policy defined on it receives a Fast Path insert or a vectored update, the hint is automatically disabled before any such operations. Disabling the hint allows auditing to occur according to the policy's terms. (One example of a Fast Path insert is the statement INSERT-WITH-APPEND-hint.)
- The audit_condition must be a boolean expression that can be evaluated using the values in the row being inserted, updated, or deleted. The expression can also use functions, such as the USER or SYS_CONTEXT functions.

  The expression must not combine conditions using operators such as AND and OR. audit_condition can be NULL (or omitted), which is interpreted as TRUE, but it cannot contain the following elements:

  – Subqueries or sequences
  – The following attributes of the USERENV namespace when accessed using the SYS_CONTEXT function:
    * CURRENT_SQL
    * CURRENT_SQL_LENGTH
    * CURRENT_BIND
  – Any use of the pseudo columns LEVEL, PRIOR, or ROWNUM.

  Specifying an audit condition of 1=1 to force auditing of all specified statements ("statement_types") affecting the specified column ("audit_column") is no longer needed

to achieve this purpose. A NULL value for audit_condition causes audit to happen even if no rows are processed, so that all actions on a table with this policy are audited.

- The audit_condition is evaluated using the privileges of the user who creates the policy.
- For the audit_condition setting, do not include functions, which execute the auditable statement on the same base table, in the audit_condition setting. For example, suppose you create a function that executes an INSERT statement on the HR.EMPLOYEES table. The policy audit_condition contains this function and it is for INSERT statements (as set by the statement_types parameter). When the policy is used, the function executes recursively until the system has run out of memory. This can raise the error ORA-1000: maximum open cursors exceeded or ORA-00036: maximum number of recursive SQL levels (50) exceeded.
- Do not issue the DBMS_FGA.ENABLE_POLICY or DBMS_FGA.DISABLE_POLICY statement from a policy function in a condition.
- The audit function (handler_module) is an alerting mechanism for the administrator. The required interface for such a function is as follows:

  ```
  PROCEDURE fname ( object_schema VARCHAR2, object_name VARCHAR2, policy_name
  VARCHAR2 )  AS ...
  ```

  where *fname* is the name of the procedure, *object_schema* is the name of the schema of the table audited, *object_name* is the name of the table to be audited, and *policy_name* is the name of the policy being enforced. The audit function will be executed with the function owner's privilege.
- Because traditional auditing is desupported, omit the audit_trail parameter because the audit records are written to the unified audit trail, viewable by querying the UNIFIED_AUDIT_TRAIL data dictionary view.
- Be aware that sensitive data, such as credit card information, can be recorded in clear text.
- You can change the operating system destination by using the following statement:

  ```
  ALTER SYSTEM SET AUDIT_FILE_DEST =  new_directory DEFERRED
  ```

  Starting with Oracle Database 23ai, the AUDIT_FILE_DEST parameter is deprecated.
- The audit_column_opts parameter establishes whether a statement is audited
  - when the query references *any* column specified in the audit_column parameter (audit_column_opts = DBMS_FGA.ANY_COLUMNS), or
  - only when *all* such columns are referenced (audit_column_opts = DBMS_FGA.ALL_COLUMNS).

  The default is DBMS_FGA.ANY_COLUMNS.

  The ALL_AUDIT_POLICIES view also shows audit_column_opts.
- When audit_column_opts is set to DBMS_FGA.ALL_COLUMNS, a SQL statement is audited only when all the columns mentioned in audit_column have been explicitly referenced in the statement. And these columns must be referenced in the same SQL-statement or in the sub-select.

  All these columns must refer to a single table/view or alias.

  If a SQL statement selects the columns from different table aliases, the statement will not be audited.

- For `SQL_TEXT` and `SQL_BIND` element values (`CLOB` type columns), the dynamic view shows only the first 4000 characters. The underlying XML file may have more than 4000 characters for such `SQL_TEXT` and `SQL_BIND` values.
- Error handling is the same as when `AUDIT_TRAIL=OS`. If any error occurs in writing an audit record, the audited operation fails and an alert message is logged.
- The policy event handler module will be executed with the module owner's privilege.
- Do not create recursive fine-grained audit handlers. For example, suppose you create a handler that executes an `INSERT` statement on the `HR.EMPLOYEES` table. The policy that is associated with this handler is for `INSERT` statements (as set by the statement_types parameter). When the policy is used, the handler executes recursively until the system has run out of memory. This can raise the error `ORA-1000: maximum open cursors exceeded` or `ORA-00036: maximum number of recursive SQL levels (50) exceeded`. See also *Oracle Database Security Guide* with regard to creating a fine-grained audit policy.
- The fine-grained audit handler module should not have explicit `COMMIT`, `ROLLBACK`, and `DDL` statements mentioned in it.

> **See Also:**
>
> *Oracle Database Security Guide* for an example of creating an email alert handler for a fine-grained audit policy

Examples

```
DBMS_FGA.ADD_POLICY (
   object_schema      =>  'scott',
   object_name        =>  'emp',
   policy_name        =>  'mypolicy1',
   audit_condition    =>  'sal < 100',
   audit_column       =>  'comm,sal',
   handler_schema     =>   NULL,
   handler_module     =>   NULL,
   enable             =>   TRUE,
   statement_types    =>  'INSERT, UPDATE',
   audit_column_opts  =>   DBMS_FGA.ANY_COLUMNS,
   policy_owner       =>  'sec_admin);
```

DISABLE_POLICY Procedure

This procedure disables an audit policy.

Syntax

```
DBMS_FGA.DISABLE_POLICY(
   object_schema  IN  VARCHAR2,
   object_name    IN  VARCHAR2,
   policy_name    IN  VARCHAR2);
```

Parameters

Table 86-3 DISABLE_POLICY Procedure Parameters

Parameter	Description
object_schema	Schema of the object to be audited. If NULL, the current schema is assumed.
object_name	Name of the object to be audited
policy_name	Unique name of the policy

The default value for object_schema is NULL. If NULL, the current schema is assumed.

Examples

```
DBMS_FGA.DISABLE_POLICY (
object_schema    =>  'scott',
object_name      =>  'emp',
policy_name      =>  'mypolicy1');
```

DROP_POLICY Procedure

This procedure drops an audit policy.

Syntax

```
DBMS_FGA.DROP_POLICY(
   object_schema  IN  VARCHAR2,
   object_name    IN  VARCHAR2,
   policy_name    IN  VARCHAR2);
```

Parameters

Table 86-4 DROP_POLICY Procedure Parameters

Parameter	Description
object_schema	Schema of the object to be audited. If NULL, the current schema is assumed.
object_name	Name of the object to be audited
policy_name	Unique name of the policy

Usage Notes

The DBMS_FGA procedures cause current DML transactions, if any, to commit before the operation unless they are inside a DDL event trigger. With DDL transactions, the DBMS_FGA procedures are part of the DDL transaction. The default value for object_schema is NULL. If NULL, the current schema is assumed.

> **Note:**
>
> Oracle Database automatically drops the audit policy if you remove the object specified in the `object_name` parameter of the `DBMS_FGA.ADD_POLICY` procedure, or if you drop the user who created the audit policy.

Examples

```
DBMS_FGA.DROP_POLICY (
object_schema   =>  'scott',
object_name     =>  'emp',
policy_name     =>  'mypolicy1');
```

ENABLE_POLICY Procedure

This procedure enables an audit policy.

Syntax

```
DBMS_FGA.ENABLE_POLICY(
   object_schema  IN  VARCHAR2,
   object_name    IN  VARCHAR2,
   policy_name    IN  VARCHAR2,
   enable         IN  BOOLEAN);
```

Parameters

Table 86-5 ENABLE_POLICY Procedure Parameters

Parameter	Description
`object_schema`	Schema of the object to be audited. If `NULL`, the current schema is assumed.
`object_name`	Name of the object to be audited
`policy_name`	Unique name of the policy
`enable`	Defaults to `TRUE` to enable the policy

Examples

```
DBMS_FGA.ENABLE_POLICY (
object_schema    =>  'scott',
object_name      =>  'emp',
policy_name      =>  'mypolicy1',
enable           =>   TRUE);
```