15

Using Transparent Sensitive Data Protection

Transparent sensitive data protection enables you to identify all table columns in a database that hold sensitive data.

- About Transparent Sensitive Data Protection
  Transparent sensitive data protection is a way to identify and label table columns that hold sensitive information.
- General Steps for Using Transparent Sensitive Data Protection
  To use Transparent Data Sensitive Data Protection (TSDP) with Oracle Data Redaction and Oracle Virtual Private Database, you must follow a set of general steps.
- Benefits of Transparent Sensitive Data Protection Policies
  Transparent sensitive data protection has several benefits.
- Privileges Required for Using Transparent Sensitive Data Protection
  To use transparent sensitive data protection, you must have the `EXECUTE` privilege for several PL/SQL packages.
- How a Multitenant Environment Affects Transparent Sensitive Data Protection
  You can apply Transparent Sensitive Data Protection (TSDP) policies to the current PDB or current application PDB only.
- Creating Transparent Sensitive Data Protection Policies
  You must create a sensitive type, find the sensitive columns to be protected, and then import these columns from Application Dependency Management (ADM) into your database.
- Altering Transparent Sensitive Data Protection Policies
  The `DBMS_TSDP_PROTECT.ALTER_POLICY` procedure can alter a TSDP policy.
- Disabling Transparent Sensitive Data Protection Policies
  The `DBMS_TSDP_PROTECT.DISABLE_PROTECTION_COLUMN` procedure disables one or all TSDP policies.
- Dropping Transparent Sensitive Data Protection Policies
  You can drop an entire TSDP policy or a condition-enable-options combination from the policy.
- Using the Predefined REDACT_AUDIT Policy for Redaction
  The predefined `REDACT_AUDIT` policy masks bind values, which can appear in trace files when an event is set.
- Transparent Sensitive Data Protection Policies with Data Redaction
  Oracle Data Redaction features work with transparent sensitive data protection policies.
- Using Transparent Sensitive Data Protection Policies with Oracle VPD Policies
  You can combine protections from TSDP and Oracle Virtual Private Database into one policy.
- Using Transparent Sensitive Data Protection Policies with Unified Auditing
  The transparent sensitive data protection and unified auditing procedures can combine the protections of these two features.

- Using Transparent Sensitive Data Protection Policies with Fine-Grained Auditing
  The transparent sensitive data protection and fine-grained auditing procedures can combine the protections of these two features.
- Using Transparent Sensitive Data Protection Policies with TDE Column Encryption
  The TSDP procedures and Transparent Data Encryption column encryption statements can combine the protections of these two features.
- Transparent Sensitive Data Protection Data Dictionary Views
  Oracle Database provides data dictionary views that list information about transparent sensitive data protection policies.

15.1 About Transparent Sensitive Data Protection

Transparent sensitive data protection is a way to identify and label table columns that hold sensitive information.

This feature enables you to quickly find the table columns in a database that hold sensitive data, classify this data, and then create a policy that protects this data as a whole for a given class. Examples of this type of sensitive data are credit card numbers or Social Security numbers.

The TSDP policy then protects the sensitive data in these table columns by using either Oracle Data Redaction or Oracle Virtual Private Database settings. The TSDP policy applies at the column level of the table that you want to protect, targeting a specific column data type, such as all `NUMBER` data types of columns that contain credit card information. You can create a uniform TSDP policy for all of the data that you classify, and then modify this policy as necessary, as compliance regulations change. Optionally, you can export the TSDP policies for use in other databases.

The benefits of TSDP policies are that you easily can create and apply TSDP policies throughout a large organization with numerous databases. This helps auditors greatly by enabling them to estimate the protection for the data that the TSDP policies target. TSDP is particularly useful for government environments, in which you may have a lot of data with similar security restrictions and you must apply a policy to all of this data consistently. The policy could be to redact it, encrypt it, control access to it, audit access to it, and mask it in the audit trail. Therefore, TSDP helps you to efficiently and consistently manage security policies across your database.

15.2 General Steps for Using Transparent Sensitive Data Protection

To use Transparent Data Sensitive Data Protection (TSDP) with Oracle Data Redaction and Oracle Virtual Private Database, you must follow a set of general steps.

1. Create a sensitive type to classify the types of columns that you want to protect.

   For example, you can create a sensitive type to classify all Social Security numbers or credit card numbers. To create the sensitive type, either use the `DBMS_TSDP_MANAGE.ADD_SENSITIVE_TYPE` PL/SQL procedure or use an Enterprise Manager Cloud Control Application Data Model. To add multiple sensitive types in one operation from an Application Data Model, you can use the `DBMS_TSDP_MANAGE.IMPORT_SENSITIVE_TYPES` procedure.

2. Identify a list of sensitive columns that are associated with the sensitive types.

   To determine and generate this list, you can use either of the following methods:

- The `DBMS_TSDP_MANAGE.ADD_SENSITIVE_COLUMN` procedure individually identifies sensitive columns.
- An Oracle Enterprise Manager Cloud Control Application Data Model enables you to identify a group of sensitive columns. It then prepares this list of sensitive columns in XML format, which you then import into your database.

3. If you used an Application Data Model for Step 2, then import the list of sensitive columns from the Application Data Model into your database by using the `DBMS_TSDP_MANAGE.IMPORT_DISCOVERY_RESULT` procedure.
4. Create the TSDP policy by using the `DBMS_TSDP_PROTECT.ADD_POLICY` procedure within an anonymous block that defines the Data Redaction or Virtual Private Database settings that you want to use.
5. Associate the TSDP policy with one or more sensitive types by using the `DBMS_TSDP_PROTECT.ASSOCIATE_POLICY` procedure.
6. Enable the TSDP policy protections by using the `DBMS_TSDP_PROTECT.ENABLE_PROTECTION_SOURCE`, `DBMS_TSDP_PROTECT.ENABLE_PROTECTION_COLUMN`, or the `DBMS_TSDP_PROTECT.ENABLE_PROTECTION_TYPE` procedure.
7. Optionally, export the TSDP policy to other databases by using Oracle Data Pump to perform a full database export. (You cannot individually export TSDP policies.)

15.3 Benefits of Transparent Sensitive Data Protection Policies

Transparent sensitive data protection has several benefits.

These benefits are as follows:

- **You configure the sensitive data protection once, and then deploy this protection as necessary.** You can configure transparent sensitive data protection policies to designate how a class of data (for example, credit card columns) must be protected without actually having to specify the target data. In other words, when you create the transparent sensitive data protection policy, you do not need to include references to the actual target columns that you want to protect. The transparent sensitive data protection policy finds these target columns based on a list of sensitive columns in the database and the policy's associations with the specified sensitive types. This can be useful when you add more sensitive data to your databases after you have created the transparent sensitive data protection policies. After you create the policy, you can enable protection for the sensitive data in a single step (for example, enable protection based on the entire source database). The sensitive type of the new data and the sensitive type and policy associations determine how the sensitive data is protected. In this way, as new sensitive data is added, you do not need to configure its protection, as long as the current policy for that data type still meets you data protection policy requirements.
- **You can manage protection of multiple sensitive columns.** You can enable or disable protection for multiple sensitive columns based on a suitable attribute (such as the source database of the identification, the sensitive type itself, or a specific schema, table, or column). This granularity provides a high level of control over data security. The design of this feature enables you to manage data security based on specific compliance needs for large data sets that fall under the purview of these compliance regulations. You can configure data security based on a specific category rather than for each individual column.
- **You can protect the sensitive columns identified using the Oracle Enterprise Manager Cloud Control Application Data Modeling (ADM) feature.** You can use the Cloud Control ADM feature to create sensitive types and discover a list of sensitive columns. Then you can import this list of sensitive columns and their corresponding

sensitive types into your database. From there, you can create and manage transparent sensitive data protection policies using this information.

15.4 Privileges Required for Using Transparent Sensitive Data Protection

To use transparent sensitive data protection, you must have the `EXECUTE` privilege for several PL/SQL packages.

These privileges are as follows:

- `DBMS_TSDP_MANAGE`, which enables you to import and manage sensitive columns and sensitive types into your database. The procedures in this package run with invoker's rights. Typically, an application database administrator will be granted privileges for this package.
- `DBMS_TSDP_PROTECT`, which you use to create the TSDP policy. The procedures in this package run with invoker's rights. Typically, a security database administrator will be granted privileges for this package.
- `DBMS_REDACT` and the `ADMINISTER REDACTION POLICY` privilege, if you plan to create Data Redaction policies. Typically, a security database administrator will be granted privileges for this package.
- `EXECUTE` privilege on the `DBMS_RLS` package and be granted the `ADMINISTER ROW LEVEL SECURITY POLICY` system privilege for administering a RLS policy in another schema than yourself, if you plan to incorporate Oracle Virtual Private Database functionality into your TSDP policies. Typically, a security database administrator will be granted privileges for this package.

For better separation of duty, these packages are designed so that either an application database administrator has control over one area of the TSDP policy creation (as in the case of the `DBMS_TSDP_MANAGE` package) or a security database administrator (for the `DBMS_TSDP_PROTECT`, `DBMS_REDACT`, and `DBMS_RLS` packages).

15.5 How a Multitenant Environment Affects Transparent Sensitive Data Protection

You can apply Transparent Sensitive Data Protection (TSDP) policies to the current PDB or current application PDB only.

If you are using Enterprise Manager Cloud Control Application Data Model, then you can find sensitive columns that belong to both local and common application objects (that is, common objects that are visible and accessible in the current PDB) inside the PDB. This enables you to use a TSDP policy to protect both local objects to the PDB and common objects that are accessible from the PDB.

In an application root:

- For application containers in general:
  - When you create scripts for application install, upgrade, patch, or uninstall operations, you can include SQL statements within the `ALTER PLUGGABLE DATABASE` *`app_name`* `BEGIN INSTALL` and `ALTER PLUGGABLE DATABASE` *`app_name`* `END INSTALL` blocks to perform various operations. If you include TSDP statements within these blocks, then

the TSDP statements will fail. You can, however, include TSDP statements outside these blocks in the script.

- In the application root:
  - You can perform TSDP operations on both application common objects and application root local objects.
  - A TSDP policy that is defined in the application root container behaves as if it is a local policy to the application root. That is, the policy is effective only in the application root container.

In an application PDB:

- The security policies that protect an application PDB apply to TSDP operations that are performed on local application objects.
- The security policies that protect an application PDB apply to TSDP operations that are performed on application common objects that are accessed from the PDB. However, access to the application common object outside the application PDB is not governed by the security policy that protects the application PDB.

You can find a listing of TSDP policies and the security features that are associated with them by querying the `DBA_TSDP_POLICY_FEATURE` data dictionary view. To find all PDBs, query the `DBA_PDBS` view.

Related Topics

- *Oracle Database Reference*

15.6 Creating Transparent Sensitive Data Protection Policies

You must create a sensitive type, find the sensitive columns to be protected, and then import these columns from Application Dependency Management (ADM) into your database.

- Step 1: Create a Sensitive Type
  The sensitive type is a class of data that you designate as sensitive.
- Step 2: Identify the Sensitive Columns to Protect
  After you define a sensitive type, you are ready to identify the columns to protect.
- Step 3: Import the Sensitive Columns List from ADM into Your Database
  Next, you are ready to import the sensitive columns list from ADM into your database.
- Step 4: Create the Transparent Sensitive Data Protection Policy
  After you have created the list of sensitive columns and imported this list into your database, you can create the transparent sensitive data protection policy.
- Step 5: Associate the Policy with a Sensitive Type
  The `DBMS_TSDP_PROTECT.ASSOCIATE_POLICY` procedure associates a TSDP policy with a sensitive type.
- Step 6: Enable the Transparent Sensitive Data Protection Policy
  You can enable the TSDP policy for the current database in a protected source, a specific table column, or a specific column type.
- Step 7: Optionally, Export the Policy to Other Databases
  You can export or import the policy to or from another database.

15.6.1 Step 1: Create a Sensitive Type

The sensitive type is a class of data that you designate as sensitive.

For example, you can create a `credit_card_num_type` sensitive type for all credit card numbers.

- To create a sensitive type, either create it from an Enterprise Manager Cloud Control Application Data Model or use the `DBMS_TSDP_MANAGE.ADD_SENSITIVE_TYPE` PL/SQL procedure.

  For example, to create the sensitive type `credit_card_num_type`:

  ```
  BEGIN
   DBMS_TSDP_MANAGE.ADD_SENSITIVE_TYPE (
    sensitive_type  => 'credit_card_num_type',
    user_comment    => 'Type for credit card columns using a number data type');
  END;
  /
  ```

  In this example:

  - `sensitive_type`: Create a name that describes the sensitive type that you want to capture. This value is case sensitive, so when you reference it later on, ensure that you use the case in which you created it. You can find existing sensitive types by querying the `DBA_SENSITIVE_COLUMN_TYPES` data dictionary view.
  - `user_comment`: Optionally, enter a description for the sensitive type.

Related Topics

- *Oracle Database PL/SQL Packages and Types Reference*
- *Oracle Database Reference*

15.6.2 Step 2: Identify the Sensitive Columns to Protect

After you define a sensitive type, you are ready to identify the columns to protect.

Oracle Enterprise Manager searches for columns of sensitive data. You can use this procedure if you know which columns are sensitive. To identify the columns to protect, based on the sensitive type that you defined, you either can use an Enterprise Manager Cloud Control Application Data Model to identify sensitive columns manually, or you can use the `DBMS_TSDP_MANAGE.ADD_SENSITIVE_COLUMN` procedure.
To remove the column from the list of sensitive columns for the database, you can use the `DBMS_TSDP_MANAGE.DROP_SENSITIVE_COLUMN` procedure.

1. Find the sensitive type that you want to use.

   For example:

   ```
   SELECT NAME FROM DBA_SENSITIVE_COLUMN_TYPES;

   NAME
   --------------------
   credit_card_num_type
   ```

2. Run the `DBMS_TSDP_MANAGE.ADD_SENSITIVE_COLUMN` procedure to associate the sensitive type with a table column. Ensure that you enter the `sensitive_type` parameter using the case in which you used to create the sensitive type.

For example:

```
BEGIN
 DBMS_TSDP_MANAGE.ADD_SENSITIVE_COLUMN(
 schema_name        => 'OE',
 table_name         => 'CUST_CC',
 column_name        => 'CREDIT_CARD',
 sensitive_type     => 'credit_card_num_type',
 user_comment       => 'Sensitive column addition of credit_card_num_type');
END;
/
```

15.6.3 Step 3: Import the Sensitive Columns List from ADM into Your Database

Next, you are ready to import the sensitive columns list from ADM into your database.

- If you had used an Application Data Model to create the list of sensitive columns, then import this list into your database by running the `DBMS_TSDP_MANAGE.IMPORT_DISCOVERY_RESULT` procedure.

  If you had used the `DBMS_TSDP_MANAGE.ADD_SENSITIVE_COLUMN` procedure to identify these columns, then you can bypass this step.

For example, to import the Cloud Control Application Data Model into the current database:

```
BEGIN
 DBMS_TSDP_MANAGE.IMPORT_DISCOVERY_RESULT (
 discovery_result      => xml_adm_result,
 discovery_source      => 'ADM_Demo');
END;
/
```

In this example:

- `discovery_result` refers to the list of sensitive columns and their associated sensitive types. This list is in XML format.
- `discovery_source` refers to the name of the Application Data Model that contains the list of sensitive columns referred by the `discovery_result` setting. You can find a list of the Application Data Models from the Data Discovery and Modeling page in Enterprise Manager Cloud Control. (To access this page, from the **Enterprise** menu, select **Quality Management**, and then **Data Discovery and Modeling**. You can find a list of the sensitive columns and their associated types in the **Sensitive Columns** tab.)

15.6.4 Step 4: Create the Transparent Sensitive Data Protection Policy

After you have created the list of sensitive columns and imported this list into your database, you can create the transparent sensitive data protection policy.

- About Creating the Transparent Sensitive Data Protection Policy
  The `DBMS_TSDP_PROTECT.ADD_POLICY` procedure creates the transparent sensitive data protection policy.
- Creating the Transparent Sensitive Data Protection Policy
  You can create a transparent sensitive data protection policy that uses a partial number data type-based partial Data Redaction policy.

- Setting the Oracle Data Redaction or Virtual Private Database Feature Options
  The TSDP feature options describe the Oracle Data Redaction or Virtual Private Database settings to use for the transparent sensitive data protection policy.
- Setting Conditions for the Transparent Sensitive Data Protection Policy
  Optionally, you can specify conditions for the transparent sensitive data protection policy.
- Specifying the DBMS_TSDP_PROTECT.ADD_POLICY Procedure
  The `DBMS_TSDP_PROTECT.ADD_POLICY` procedure names the TSDP policy and executes the `FEATURE_OPTIONS` and `POLICY_CONDITIONS` settings.

15.6.4.1 About Creating the Transparent Sensitive Data Protection Policy

The `DBMS_TSDP_PROTECT.ADD_POLICY` procedure creates the transparent sensitive data protection policy.

After you have identified the sensitive columns, and if you had used an Application Data Model to create the list of sensitive columns, and imported this list into your database, you are ready to create the transparent sensitive data protection policy. To create the transparent sensitive data protection policy, you must configure it for the Virtual Private Database or Oracle Data Redaction settings that you want to use, and then apply these settings to a transparent sensitive data protection policy defined by `DBMS_TSDP_PROTECT.ADD_POLICY`.

You can create the policy by defining an anonymous block that has the following components:

- If you are using Oracle Data Redaction for your policy, a specification of the type of Data Redaction that you want to use, such as partial Data Redaction
- If you are using Oracle Virtual Private Database for your policy, a specification of the VPD settings that you want to use
- Conditions to test when the policy is enabled. For example, the data type of the column which should be satisfied before the policy can be enabled.
- A named transparent sensitive data protection policy to tie these components together, by using the `DBMS_TSDP_PROTECT.ADD_POLICY` procedure

After you create the sensitive type, it resides in the `SYS` schema.

Related Topics

- Tutorial: Creating a TSDP Policy That Uses Virtual Private Database Protection
  This tutorial demonstrates how to incorporate Oracle Virtual Private Database protection with a transparent sensitive data protection policy.

15.6.4.2 Creating the Transparent Sensitive Data Protection Policy

You can create a transparent sensitive data protection policy that uses a partial number data type-based partial Data Redaction policy.

Example 15-1 shows how to create this type of policy.

- To create the policy, use the `DBMS_TSDP_PROTECT.ADD_POLICY` procedure, as shown in Example 15-1.

Example 15-1 Creating a Transparent Sensitive Data Protection Policy

```
DECLARE
  redact_feature_options DBMS_TSDP_PROTECT.FEATURE_OPTIONS;
  policy_conditions DBMS_TSDP_PROTECT.POLICY_CONDITIONS;
BEGIN
```

```
  redact_feature_options ('expression') :=
   'SYS_CONTEXT(''USERENV'',''SESSION_USER'') =''APPUSER''';
  redact_feature_options ('function_type') := 'DBMS_REDACT.PARTIAL';
  redact_feature_options ('function_parameters') := '0,1,6';
  policy_conditions(DBMS_TSDP_PROTECT.DATATYPE) := 'NUMBER';
  policy_conditions(DBMS_TSDP_PROTECT.LENGTH) := '16';
 DBMS_TSDP_PROTECT.ADD_POLICY ('redact_partial_cc',
  DBMS_TSDP_PROTECT.REDACT,redact_feature_options,
   policy_conditions);
END;
/
```

In this example:

- `redact_feature_options DBMS_TSDP_PROTECT.FEATURE_OPTIONS` creates the variable `redact_feature_options`, which uses the `FEATURE_OPTIONS` procedure. See Setting the Oracle Data Redaction or Virtual Private Database Feature Options for more information.
- `policy_conditions DBMS_TSDP_PROTECT.POLICY_CONDITIONS` creates the variable `policy_conditions`, which uses the `POLICY_CONDITIONS` procedure. See Setting Conditions for the Transparent Sensitive Data Protection Policy for more information.
- `redact_feature_options` lines (3) write the Data Redaction policy settings to the `redact_feature_options` variable. This example applies the Data Redaction policy to the user `APPUSER` and defines the policy as a partial data redaction for number data types. See *Oracle Database Advanced Security Guide* for information about how the `function_parameters` parameter works for this case.
- `policy_conditions` lines (2) write the TSDP policy conditions to the `policy_conditions` variable (that is, the data type and length) for the protected `NUMBER` data type column.
- `DBMS_TSDP_PROTECT.ADD_POLICY` executes the `DBMS_TSDP_PROTECT.ADD_POLICY` procedure, which creates the `redact_partial_cc` TSDP policy. See Specifying the DBMS_TSDP_PROTECT.ADD_POLICY Procedure for more information.

If you want to see an example of a similar policy for VPD, see Step 4: Create and Enable a Transparent Sensitive Data Protection Policy.

15.6.4.3 Setting the Oracle Data Redaction or Virtual Private Database Feature Options

The TSDP feature options describe the Oracle Data Redaction or Virtual Private Database settings to use for the transparent sensitive data protection policy.

- For Data Redaction, define the feature options by using the name `redact_feature_options` variable and for the type, you must use the type `DBMS_TSDP_PROTECT.FEATURE_OPTIONS`, which is an associative array of the data type `VARCHAR2(TSDP_PARAM_MAX)`. Initialize these options with the parameter-value pairs that correspond with the `DBMS_REDACT.ADD_POLICY` parameters.

  For example, to specify a TSDP policy that specifies when Data Redaction should be applied:

  ```
  redact_feature_option ('expression') := 'expression';
  ```

  For a partial Data Redaction policy that uses a number data type for the protected column, the following example specifies the following additional parameter-value pairs:

```
redact_feature_options ('function_type') := 'DBMS_REDACT.PARTIAL';
redact_feature_options ('function_parameters') := 'values';
```

Similarly, for Virtual Private Database, you use the `vpd_feature_options` variable to define the VPD feature options. For example:

```
vpd_feature_options ('statement_types') := 'SELECT, INSERT, UPDATE, DELETE';
```

Related Topics

- *Oracle Database Advanced Security Guide*
- DBMS_RLS.ADD_POLICY Parameters That Are Used for TSDP Policies
  Oracle Database provides a set of parameters for fine-tuning the behavior of TSDP policies.

15.6.4.4 Setting Conditions for the Transparent Sensitive Data Protection Policy

Optionally, you can specify conditions for the transparent sensitive data protection policy.

- Specify the transparent sensitive data protection policy conditions in the following ways:
  - To define the conditions, use the name `policy_conditions` for the variable and for the type, use type `DBMS_TSDP_PROTECT.POLICY_CONDITIONS`, which is an associative array of the data type `VARCHAR2(TSDP_PARAM_MAX)`. The target column's properties should satisfy all the condition properties for the corresponding `DBMS_TSDP_PROTECT.FEATURE_OPTIONS` settings to be applied on the column.
    For example:

    ```
    policy_conditions(DBMS_TSDP_PROTECT.DATATYPE) := 'NUMBER';
    policy_conditions(DBMS_TSDP_PROTECT.LENGTH) := '16';
    ```

  - Optionally, to specify one or more of the following keys for the `POLICY_CONDITIONS` settings:
    - `DBMS_TSDP_PROTECT.DATATYPE` enables you to specify a data type.
    - `DBMS_TSDP_PROTECT.LENGTH` enables you to specify a data type length for the `DBMS_TSDP_PROTECT.DATATYPE` key.
    - `DBMS_TSDP_PROTECT.PARENT_SCHEMA` enables you to restrict the policy to a specific schema. If you omit this setting, then the policy applies to all schemas in the database.
    - `DBMS_TSDP_PROTECT.PARENT_TABLE` enables you to restrict the policy to a table specified by the `DBMS_TSDP_PROTECT.PARENT_SCHEMA` key. If you omit this setting, then the policy applies to all tables within the specified schema.
  - If you choose to omit conditions, you still must include the following line in the `DECLARE` variables. (In this case, the default value for `policy_conditions` is an empty associative array.)

    ```
    policy_conditions SYS.DBMS_TSDP_PROTECT.POLICY_CONDITIONS;
    ```

15.6.4.5 Specifying the DBMS_TSDP_PROTECT.ADD_POLICY Procedure

The `DBMS_TSDP_PROTECT.ADD_POLICY` procedure names the TSDP policy and executes the `FEATURE_OPTIONS` and `POLICY_CONDITIONS` settings.

In the policy, the `redact_feature_options` and the `policy_conditions` settings work together: When the policy is enabled (using any of the `DBMS_TSDP_PROTECT.ENABLE_PROTECTION`*

procedures) on the target object, then the `redact_feature_options` settings apply only if the corresponding `policy_condition` settings are satisfied.

- To specify a procedure that names the transparent sensitive data protection policy and executes the necessary settings, include the following parameters:
  - `policy_name` creates a name for the TSDP policy. The name that you enter is stored in the database using the case sensitivity that you used when you created it. For example, if you had entered `redact_partial_cc`, then the database stores it as `redact_partial_cc`, not `redact_partial_cc`.
  - `security_feature` refers to the security feature the TSDP policy will use. Enter `DBMS_TSDP_PROTECT.REDACT` to specify Oracle Data Redaction.
  - `policy_enable_options` refers to the variable that you defined for the `DBMS_TSDP_PROTECT.FEATURE_OPTIONS` type.
  - `policy_apply_condition` refers to the variable that you defined for the `DBMS_TSDP_PROTECT.POLICY_CONDITIONS` type.

  For example:

```
DBMS_TSDP_PROTECT.ADD_POLICY('redact_partial_cc', DBMS_TSDP_PROTECT.REDACT,
redact_feature_options, policy_conditions);
```

15.6.5 Step 5: Associate the Policy with a Sensitive Type

The `DBMS_TSDP_PROTECT.ASSOCIATE_POLICY` procedure associates a TSDP policy with a sensitive type.

1. Find the sensitive type that you want to use.

   For example, to find a list of all sensitive types:

```
SELECT NAME FROM DBA_SENSITIVE_COLUMN_TYPES ORDER BY NAME;

NAME
--------------------
credit_card_num_type
```

2. Run the `DBMS_TSDP_PROTECT.ASSOCIATE_POLICY` procedure to associate the policy with a sensitive column type.

   For example:

```
BEGIN
 DBMS_TSDP_PROTECT.ASSOCIATE_POLICY(
 policy_name        => 'redact_partial_cc',
 sensitive_type     => 'credit_card_num_type',
 associate          => true);
END;
/
```

   The following query shows that the `credit_card_num_type` is now associated with the `redact_partial_cc` policy.

```
SELECT POLICY_NAME, SENSITIVE_TYPE FROM DBA_TSDP_POLICY_TYPE ORDER BY SENSITIVE_TYPE;

POLICY_NAME       SENSITIVE_TYPE
----------------- --------------------
redact_partial_cc  credit_card_num_type
```

15.6.6 Step 6: Enable the Transparent Sensitive Data Protection Policy

You can enable the TSDP policy for the current database in a protected source, a specific table column, or a specific column type.

- Enabling Protection for the Current Database in a Protected Source
  You can enable transparent sensitive data protection for the current database in a protected source.
- Enabling Protection for a Specific Table Column
  You can enable transparent sensitive data protection for a specific column in a table.
- Enabling Protection for a Specific Column Type
  You can enable transparent sensitive data protection for a specific column type, such as all columns that use the `VARCHAR2` data type.

15.6.6.1 Enabling Protection for the Current Database in a Protected Source

You can enable transparent sensitive data protection for the current database in a protected source.

If you must disable the protection, then you can run the `DBMS_TSDP_PROTECT.DISABLE_PROTECTION_SOURCE` procedure.

- Run the `DBMS_TSDP_PROTECT.ENABLE_PROTECTION_SOURCE` procedure to enable this type of protection.

For example, to enable transparent sensitive data protection policies for the `orders_db` database.

```
BEGIN
 DBMS_TSDP_PROTECT.ENABLE_PROTECTION_SOURCE(
  discovery_source       => 'orders_db');
END;
/
```

15.6.6.2 Enabling Protection for a Specific Table Column

You can enable transparent sensitive data protection for a specific column in a table.

Remember that you can enable only one policy per table. If you must disable the protection, then you can run the `DBMS_TSDP_PROTECT.DISABLE_PROTECTION_COLUMN` procedure.

- Run the `DBMS_TSDP_PROTECT.ENABLE_PROTECTION_COLUMN` procedure to enable this type of protection.

For example, to enable the transparent sensitive data protection policy `redact_partial_cc` for a specific table column:

```
BEGIN
 DBMS_TSDP_PROTECT.ENABLE_PROTECTION_COLUMN(
  schema_name          => 'OE',
  table_name           => 'CUST_CC',
  column_name          => 'CREDIT_CARD',
  policy               => 'redact_partial_cc');
END;
/
```

If an `ORA-45622: warnings generated during policy enforcement` error appears, then check the configuration of the policy. In this example, the `redact_partial_cc` policy is enabled on a column if this column is of the `NUMBER` data type and has a length of `16`. Even though the `OE.CUST_CC.CREDIT_CARD` column is associated with the `redact_partial_cc` policy, the policy is not enabled if this column fails to satisfy the conditions (data type and length).

15.6.6.3 Enabling Protection for a Specific Column Type

You can enable transparent sensitive data protection for a specific column type, such as all columns that use the `VARCHAR2` data type.

If you must disable the protection, then you can run the `DBMS_TSDP_PROTECT.DISABLE_PROTECTION_TYPE` procedure.

- Run the `DBMS_TSDP_PROTECT.ENABLE_PROTECTION_TYPE` procedure to enable this type of protection.

For example, to enable transparent sensitive data protection for all columns that use the `credit_card_num_type` sensitive type:

```
BEGIN
 DBMS_TSDP_PROTECT.ENABLE_PROTECTION_TYPE(
  sensitive_type           => 'credit_card_num_type');
END;
/
```

15.6.7 Step 7: Optionally, Export the Policy to Other Databases

You can export or import the policy to or from another database.

- To export or import the TSDP policy to or from another database, use Oracle Data Pump to perform a full export or import of the database that contains the policy.

  Remember that the export and import operations apply to the entire database, not just the transparent sensitive data protection policy.

Related Topics

- *Oracle Database Utilities*
- Using *Oracle Database Vault Administrator's Guide*

15.7 Altering Transparent Sensitive Data Protection Policies

The `DBMS_TSDP_PROTECT.ALTER_POLICY` procedure can alter a TSDP policy.

When you alter a transparent data protection policy, you must define how the Data Redaction settings must change, and then apply these changes to the transparent sensitive data protection policy itself. You can find a list of existing policies and their protection definitions by querying the `DBA_TSDP_POLICY_FEATURE` data dictionary view.

- To alter a transparent sensitive data protection policy, use the `DBMS_TSDP_PROTECT.ALTER_POLICY` procedure.

  For example, to alter an existing transparent sensitive data protection policy:

  ```
  DECLARE
    redact_feature_options SYS.DBMS_TSDP_PROTECT.FEATURE_OPTIONS;
    policy_conditions SYS.DBMS_TSDP_PROTECT.POLICY_CONDITIONS;
    BEGIN
  ```

```
  redact_feature_options ('expression') :=
   'SYS_CONTEXT(''USERENV'',''SESSION_ USER'') ='''APPUSER''';
  redact_feature_options ('function_type') := 'DBMS_REDACT.PARTIAL';
  redact_feature_options ('function_parameters') := '9,1,6';
  policy_conditions(DBMS_TSDP_PROTECT.DATATYPE) := 'NUMBER';
  policy_conditions(DBMS_TSDP_PROTECT.LENGTH) := '22';
 DBMS_TSDP_PROTECT.ALTER_POLICY ('redact_partial_cc',
  redact_feature_options, policy_conditions);
END;
/
```

In this example:

- `redact_feature_options SYS.DBMS_TSDP_PROTECT.FEATURE_OPTIONS` creates the variable `redact_feature_options`, which uses the `FEATURE_OPTIONS` data type.
- `policy_conditions SYS.DBMS_TSDP_PROTECT.POLICY_CONDITIONS` creates the variable `policy_conditions`, which uses the `POLICY_CONDITIONS` data type.
- `redact_feature_options ... redact_feature_options` writes the Data Redaction policy settings to the `redact_feature_option` variable. This example applies the Data Redaction policy to the user `APPUSER`, defines the policy as a partial data redaction for number data types.
- `policy_conditions ... policy_conditions` writes the TSDP policy conditions to the `policy_conditions` variable (that is, the data type and length) for the protected `NUMBER` data type column.
- `DBMS_TSDP_PROTECT.ALTER_POLICY ...` executes the `DBMS_TSDP_PROTECT.ALTER_POLICY` procedure, which alters the `redact_partial_cc` TSDP policy to use the definitions set in the `redact_feature_options` and `policy_conditions` variables.

15.8 Disabling Transparent Sensitive Data Protection Policies

The `DBMS_TSDP_PROTECT.DISABLE_PROTECTION_COLUMN` procedure disables one or all TSDP policies.

1. Query the `DBA_TSDP_POLICY_PROTECTION` data dictionary view to find the protected columns and their associated transparent sensitive data protection policies.

   For example:

   ```
   SELECT COLUMN_NAME, TSDP_POLICY FROM DBA_TSDP_POLICY_PROTECTION WHERE TABLE_NAME =
   'CUST_CC';

   COLUMN_NAME   TSDP_POLICY
   ------------  ------------------
   CREDIT_CARD   redact_partial_cc
   ```

2. Run the `DBMS_TSDP_PROTECT.DISABLE_PROTECTION_COLUMN` procedure.

   For example, to disable the `redact_partial_cc` policy on the `CREDIT_CARD` column of the `CUST_CC` table:

   ```
   BEGIN
    DBMS_TSDP_PROTECT.DISABLE_PROTECTION_COLUMN(
     schema_name          => 'OE',
     table_name           => 'CUST_CC',
     column_name          => 'CREDIT_CARD',
     policy               => 'redact_partial_cc');
   ```

```
END;
/
```

You can use the `%` wildcard in this procedure to specify multiple items. For example, to disable protection for any columns that begin with `CREDIT`, you could enter the following:

```
BEGIN
 DBMS_TSDP_PROTECT.DISABLE_PROTECTION_COLUMN(
  schema_name          => 'OE',
  table_name           => 'CUST_CC',
  column_name          => 'CREDIT%',
  policy               => 'redact_partial_cc');
END;
/
```

To disable all transparent sensitive data protection policies for a table, you can omit the `policy` parameter. For example:

```
BEGIN
 DBMS_TSDP_PROTECT.DISABLE_PROTECTION_COLUMN(
  schema_name          => 'OE',
  table_name           => 'CUST_CC',
  column_name          => '%');
END;
/
```

15.9 Dropping Transparent Sensitive Data Protection Policies

You can drop an entire TSDP policy or a condition-enable-options combination from the policy.

If the policy only has one condition-enable-options combination, then Oracle Database drops the entire policy. You do not need to disable a policy before dropping it, but you do need to drop its associated sensitive column first, then its sensitive type.

1. Query the `POLICY_NAME` column of the `DBA_TSDP_POLICY_FEATURE` data dictionary view to find the policy that you want to drop.

   ```
   SELECT POLICY_NAME FROM DBA_TSDP_POLICY_FEATURE;

   POLICY_NAME
   -----------------
   redact_partial_cc
   ```

   Remember that you must be granted the `SELECT_CATALOG_ROLE` role to query the transparent sensitive data protection data dictionary views.

2. Find the sensitive column that is associated with this policy.

   For example:

   ```
   SELECT COLUMN_NAME FROM DBA_TSDP_POLICY_PROTECTION WHERE TSDP_POLICY =
   'redact_partial_cc';

   COLUMN_NAME
   -----------------
   CREDIT_CARD
   ```

3. Drop this sensitive column.

   For example:

   ```
   BEGIN
    DBMS_TSDP_MANAGE.DROP_SENSITIVE_COLUMN (
   ```

```
   schema_name        => 'OE',
   table_name         => 'CUST_CC',
   column_name        => 'CREDIT_CARD');
END;
/
```

4. Find the sensitive type that is associated with this policy.

   For example:

```
SELECT SENSITIVE_TYPE FROM DBA_TSDP_POLICY_TYPE WHERE POLICY_NAME =
'redact_partial_cc';

SENSITIVE_TYPE
--------------------
credit_card_num_type
```

5. Drop this sensitive type.

   For example:

```
BEGIN
 DBMS_TSDP_MANAGE.DROP_SENSITIVE_TYPE (   sensitive_type     =>
'credit_card_num_type');END;
/
```

6. Run the `DBMS_TSDP_PROTECT.DROP_POLICY` procedure to drop the policy.

   For example, to completely drop the policy:

```
BEGIN
 DBMS_TSDP_PROTECT.DROP_POLICY(
   policy_name     => 'redact_partial_cc');
END;
/
```

   To drop the default condition-enable options combination from the policy:

```
DECLARE
   policy_conditions DBMS_TSDP_PROTECT.POLICY_CONDITIONS;
BEGIN
   DBMS_TSDP_PROTECT.DROP_POLICY ('redact_partial_cc', policy_conditions);
END;
/
```

   To drop the default condition-enable options combination from the policy based on a specific condition:

```
DECLARE
   policy_conditions DBMS_TSDP_PROTECT.POLICY_CONDITIONS;
BEGIN
   policy_conditions (DBMS_TSDP_PROTECT.DATATYPE) := 'NUMBER';
   DBMS_TSDP_PROTECT.DROP_POLICY ('redact_partial_cc', policy_conditions);
END;
/
```

15.10 Using the Predefined REDACT_AUDIT Policy for Redaction

The predefined `REDACT_AUDIT` policy masks bind values, which can appear in trace files when an event is set.

- About the REDACT_AUDIT Policy
  The predefined REDACT_AUDIT transparent sensitive data protection policy masks bind values.
- Variables Associated with Sensitive Columns
  Bind variables affect the use of sensitive columns with conditions, SELECT items, and INSERT or UPDATE operations.
- How Bind Variables on Sensitive Columns Behave with Views
  A bind variable that appears in a query on a view is considered sensitive if the view column references a sensitive column.
- Disabling the REDACT_AUDIT Policy
  By default, the REDACT_AUDIT policy is enabled for all sensitive columns.
- Enabling the REDACT_AUDIT Policy
  You can enable the REDACT_AUDIT policy for a specific sensitive column or for all columns in the database.

15.10.1 About the REDACT_AUDIT Policy

The predefined REDACT_AUDIT transparent sensitive data protection policy masks bind values.

The bind values of the bind variables that are used in SQL statements can appear in audit records when auditing is configured. Similarly, bind values can appear in trace files when the appropriate event is set. Bind values can also appear when you query the V$SQL_BIND_DATA dynamic view.

The REDACT_AUDIT transparent sensitive data protection policy displays the data as an asterisk (*) in audit records, trace files, and in V$SQL_BIND_DATA view queries. By default the REDACT_AUDIT policy is associated with every sensitive type in the database. When you identify a column as sensitive, by default, the REDACT_AUDIT policy is enabled for it.

You can disable and enable the REDACT_AUDIT policy, but you cannot alter or drop it.

15.10.2 Variables Associated with Sensitive Columns

Bind variables affect the use of sensitive columns with conditions, SELECT items, and INSERT or UPDATE operations.

- About Variables Associated with Sensitive Columns
  You can associate variables with sensitive columns in TSDP policies.
- Bind Variables and Sensitive Columns in the Expressions of Conditions
  You can include sensitive columns in SQL queries that have WHERE clauses.
- A Bind Variable and a Sensitive Column Appearing in the Same SELECT Item
  If a column in a SELECT item is sensitive, then all the binds in the SELECT item are considered sensitive.
- Bind Variables in Expressions Assigned to Sensitive Columns in INSERT or UPDATE Operations
  You can assign multiple bind variables to different columns in one INSERT or UPDATE statement.

15.10.2.1 About Variables Associated with Sensitive Columns

You can associate variables with sensitive columns in TSDP policies.

A bind variable can be considered to be sensitive or "associated" with a sensitive column if the bind variable occurs in the same comparison condition as a sensitive column, if it occurs in a `SELECT` statement alongside a sensitive column, or if it occurs in an `INSERT` or `UPDATE` operation that involves a sensitive column.

15.10.2.2 Bind Variables and Sensitive Columns in the Expressions of Conditions

You can include sensitive columns in SQL queries that have `WHERE` clauses.

A SQL query that contains a `WHERE` clause can include sensitive columns and bind variables for use with comparison operators such as `=`, `IS`, `IS NOT`, `LIKE`, `BETWEEN`, and `IN`, as well as in subqueries.

In the following comparison query, the bind value in `VAR1` is masked because `VAR1` and the sensitive column `SALARY` appear in the expression that is compared using the comparison condition `>`.

```
SELECT EMPLOYEE_ID FROM HR.EMPLOYEES WHERE SALARY > :VAR1;
```

In the next query, the bind values in `VAR1` and `VAR2` are masked because `VAR1`, `VAR2`, and the sensitive column `SALARY` appear in the expression that uses the comparison equality condition `=`.

```
SELECT EMPLOYEE_ID FROM HR.EMPLOYEES WHERE SALARY + :VAR1 = TO_NUMBER(:VAR2, '9G999D99');
```

For floating point conditions, the sensitive column and the bind variable appear in the expression that is evaluated. In the following example, the bind value in `VAR1` is masked because `VAR1` and the sensitive column `SALARY` appear in the expression for the `IS NOT NAN` condition.

```
SELECT COUNT( ) FROM HR.EMPLOYEES WHERE (SALARY * :VAR1) IS NOT NAN;
```

In pattern matching conditions, the sensitive column and the bind variable appear as arguments. In the following example, the bind value in `VAR1` is masked because `VAR1` and the sensitive column `LAST_NAME` are the arguments for the `LIKE` condition.

```
SELECT LAST_NAME FROM HR.EMPLOYEES WHERE LAST_NAME LIKE :VAR1;
```

For `BETWEEN` conditions, the sensitive column and the bind variable appear in the expressions that are arguments. In the following example, bind values in `VAR1` and `VAR2` are masked because `VAR1`, `VAR2`, and `SALARY` appear in expressions that are arguments to the `BETWEEN` condition.

```
SELECT EMPLOYEE_ID FROM HR.EMPLOYEES WHERE SALARY BETWEEN :VAR1 AND :VAR2;
```

In the next example, the sensitive column and the bind variable are the arguments of the `IN` condition. Here, the bind values in `VAR1` and `VAR2` are masked because `VAR1`, `VAR2`, and the sensitive column `SALARY` appear as arguments to the `IN` condition.

```
SELECT COUNT( ) FROM HR.EMPLOYEES WHERE SALARY IN ( :VAR1, :VAR2);
```

When a condition has a nested subquery as an argument, the bind variables and sensitive columns that appear in the nested subquery are not considered to be associated with the condition. In the following query, the sensitive column `SALARY` and the subquery are expressions for the greater-than condition `>`.

```
SELECT EMPLOYEE_ID FROM HR.EMPLOYEES WHERE SALARY > (SELECT SALARY FROM HR.EMPLOYEES
WHERE MANAGER_ID = :VAR1);
```

However, variable VAR1 is associated with column MANAGER_ID as variable VAR1 and MANAGER_ID appears in expressions being compared using the condition =. Because MANAGER_ID is not a sensitive column, variable VAR1 is not considered sensitive. The variable VAR1 is not considered to be associated with the sensitive column SALARY.

In the case of the logical conditions, model conditions, multiset conditions, XML conditions, compound conditions, IS OF type conditions, and EXISTS conditions, there can be no cases where a bind variable and a sensitive column are associated with each other. This is due to the structure or the nature of these conditions.

15.10.2.3 A Bind Variable and a Sensitive Column Appearing in the Same SELECT Item

If a column in a SELECT item is sensitive, then all the binds in the SELECT item are considered sensitive.

For example, assume that HR.EMPLOYEES.SALARY and HR.EMPLOYEES.COMMISSION_PCT are sensitive columns. In the following query, the bind variable VAR1 is considered sensitive because it appears in the same SELECT item as the sensitive column SALARY, so its bind value is masked.

```
SELECT (SALARY * :VAR1) AS BONUS AS FROM HR.EMPLOYEES WHERE EMPLOYEE_ID = :VAR2;
```

In the next example, the bind variable VAR1 is considered sensitive because it appears in the same SELECT item as SALARY. VAR2 is considered sensitive because it appears in the same SELECT item as the sensitive column COMMISSION_PCT.

```
SELECT (SALARY * :VAR1), (COMMISSION_PCT * :VAR2), (EMPNO + :VAR3) AS BONUS AS FROM
PAYROLL.ACCOUNT;
```

15.10.2.4 Bind Variables in Expressions Assigned to Sensitive Columns in INSERT or UPDATE Operations

You can assign multiple bind variables to different columns in one INSERT or UPDATE statement.

Consider the following INSERT statement:

```
INSERT INTO PAYROLL.ACCOUNT (ACCOUNT_NUM, SALARY) VALUES (:VAR1 * :VAR2 , :VAR3);
```

In this INSERT statement, the following takes place:

- The bind variables VAR1 and VAR2 appear in the expression (:VAR1 * :VAR2), which is assigned to the sensitive column ACCOUNT_NUM.
- The bind variable VAR3 is assigned to sensitive column SALARY.

Consider the following UPDATE statement:

```
UPDATE PAYROLL.ACCOUNT SET ACCOUNT_NUM = :VAR1, SALARY = :VAR2;
```

In this UPDATE statement, the following takes place:

- The bind variable VAR1 is assigned to sensitive column ACCOUNT_NUM.
- The bind variable VAR2 is assigned to sensitive column SALARY.

15.10.3 How Bind Variables on Sensitive Columns Behave with Views

A bind variable that appears in a query on a view is considered sensitive if the view column references a sensitive column.

For example, suppose you identify the `SALARY` column in the `HR.EMPLOYEES` table as sensitive. Then you create the view `EMPLOYEES_VIEW` as follows:

```
CREATE OR REPLACE VIEW HR.EMPLOYEES_VIEW AS SELECT * FROM HR.EMPLOYEES;
```

When a user references the `SALARY` column from this view in a SQL statement, any bind variable that has been associated with the `SALARY` column is considered sensitive and its bind value then masked.

```
SELECT EMPLOYEE_ID FROM HR.EMPLOYEES_VIEW WHERE SALARY = :VAR1;
```

In this case, the bind variable `VAR1` is masked because it is associated with the `HR.EMPLOYEES_VIEW.SALARY` column, which references the sensitive column `HR.EMPLOYEES.SALARY`.

15.10.4 Disabling the REDACT_AUDIT Policy

By default, the `REDACT_AUDIT` policy is enabled for all sensitive columns.

You can disable it for a specific sensitive column or all sensitive columns, and when needed, re-enable it. Remember that you cannot alter or delete the `REDACT_AUDIT` policy.

- To disable the `REDACT_AUDIT` policy, use the `DBMS_TSDP_PROTECT.DISABLE_PROTECTION_COLUMN` procedure.

For example, to disable the `REDACT_AUDIT` policy for the `SALARY` column of `HR.EMPLOYEES`:

```
BEGIN
 DBMS_TSDP_PROTECT.DISABLE_PROTECTION_COLUMN(
  schema_name          => 'HR',
  table_name           => 'EMPLOYEES',
  column_name          => 'SALARY',
  policy               => 'REDACT_AUDIT');
END;
/
```

The following example shows how to disable the `REDACT_AUDIT` policy for all sensitive columns in the current database.

```
BEGIN
 DBMS_TSDP_PROTECT.DISABLE_PROTECTION_COLUMN(
  policy               => 'REDACT_AUDIT');
END;
/
```

15.10.5 Enabling the REDACT_AUDIT Policy

You can enable the `REDACT_AUDIT` policy for a specific sensitive column or for all columns in the database.

- To enable the `REDACT_AUDIT` policy, use the `DBMS_TSDP_PROTECT.ENABLE_PROTECTION_COLUMN` procedure.

For example, to re-enable the `REDACT_AUDIT` policy for the `SALARY` column of `HR.EMPLOYEES`:

```
BEGIN
 DBMS_TSDP_PROTECT.ENABLE_PROTECTION_COLUMN(
  schema_name          => 'HR',
  table_name           => 'EMPLOYEES',
  column_name          => 'SALARY',
  policy               => 'REDACT_AUDIT');
END;
/
```

The following example shows how to enable the `REDACT_AUDIT` policy for all sensitive columns in the current database.

```
BEGIN
 DBMS_TSDP_PROTECT.ENABLE_PROTECTION_COLUMN(
  policy               => 'REDACT_AUDIT');
END;
/
```

15.11 Transparent Sensitive Data Protection Policies with Data Redaction

Oracle Data Redaction features work with transparent sensitive data protection policies.

The Data Redaction function types, function parameters, and expressions can be used in the TSDP policy definition. For example, you can set the enable the TSDP policy to use `FULL` or `PARTIAL` data redaction. This chapter uses Data Redaction for examples of managing TSDP policies.

Related Topics

- Creating Transparent Sensitive Data Protection Policies
  You must create a sensitive type, find the sensitive columns to be protected, and then import these columns from Application Dependency Management (ADM) into your database.
- *Oracle Database Advanced Security Guide*

15.12 Using Transparent Sensitive Data Protection Policies with Oracle VPD Policies

You can combine protections from TSDP and Oracle Virtual Private Database into one policy.

- About Using TSDP Policies with Oracle Virtual Private Database Policies
  To incorporate Oracle Virtual Private Database protection with transparent sensitive data protection policies, you must use the `DBMS_TSDP_PROTECT` and `DBMS_RLS` packages.
- DBMS_RLS.ADD_POLICY Parameters That Are Used for TSDP Policies
  Oracle Database provides a set of parameters for fine-tuning the behavior of TSDP policies.
- Tutorial: Creating a TSDP Policy That Uses Virtual Private Database Protection
  This tutorial demonstrates how to incorporate Oracle Virtual Private Database protection with a transparent sensitive data protection policy.

15.12.1 About Using TSDP Policies with Oracle Virtual Private Database Policies

To incorporate Oracle Virtual Private Database protection with transparent sensitive data protection policies, you must use the `DBMS_TSDP_PROTECT` and `DBMS_RLS` packages.

This feature works as follows:

1. You create a VPD policy function with a suitable predicate. Later on, when you create the TSDP policy, you will refer to this VPD policy function by using the `policy_function` setting of the `DBMS_RLS.ADD_POLICY` procedure for the `feature_options` parameter of the `DBMS_TSDP_PROTECT.ADD_POLICY` procedure.
2. You create a TSDP policy with the necessary VPD settings similar to the VPD policy function.

   The TSDP policy uses parameter settings from the `DBMS_RLS.ADD_POLICY` procedure to provide VPD protection. Be aware that parameters from the `DBMS_RLS.ADD_GROUPED_POLICY` policy are not supported.
3. You associate the TSDP policy with the necessary sensitive types by using the `DBMS_TSDP_PROTECT.ASSOCIATE_POLICY` procedure.
4. You then enable TSDP protection by using any of the `DBMS_TSDP_PROTECT.ENABLE_PROTECTION_*` procedures.
5. You enable the TSDP policy. At this point, Oracle Database creates an internal VPD policy that uses the function that you created.

   The name of the internal policy begins with `ORA$VPD` followed by an identifier (for example, `ORA$VPD_6J6L3RSJSN2VAN0XF)`. You can find this policy by querying the `POLICY_NAME` column of the `DBA_POLICIES` data dictionary view.
6. When users query the table, the output for the column is based on both the VPD protections and the TSDP policy that are now in place.
7. These protections remain in place until you disable the TSDP policy for this column. At that point, Oracle Database automatically drops the internal VPD policy, because it is no longer needed. If you reenable the TSDP policy, then the internal VPD policy is recreated.

Related Topics

- DBMS_RLS.ADD_POLICY Parameters That Are Used for TSDP Policies
  Oracle Database provides a set of parameters for fine-tuning the behavior of TSDP policies.
- Function to Generate the Dynamic WHERE Clause
  The Oracle Virtual Private Database (VPD) function defines the restrictions that you want to enforce.

15.12.2 DBMS_RLS.ADD_POLICY Parameters That Are Used for TSDP Policies

Oracle Database provides a set of parameters for fine-tuning the behavior of TSDP policies.

Table 15-1 describes the `DBMS_RLS.ADD_POLICY` parameters that are permissible in the `FEATURE_OPTIONS` parameter when you use the `DBMS_TSDP_PROTECT.ADD_POLICY` or `DBMS_TSDP_PROTECT.ALTER_POLICY` procedure.

Table 15-1 DBMS_RLS.ADD_POLICY Parameters Used for TSDP Policies

Parameter	Description	Default
function_schema	Schema of the policy function (current default schema, if NULL). If no function_schema is specified, then the current user's schema is assumed.	NULL
policy_function	Name of a function that generates a predicate for the policy. If the function is defined within a package, then you must include the name of the package (for example, my_package.my_function).	NULL
statement_types	Statement types to which the policy applies. It can be any combination of INDEX, SELECT, INSERT, UPDATE, or DELETE. The default is to apply to most of these types except INDEX.	NULL
update_check	Optional argument for INSERT or UPDATE statement types. Setting update_check to TRUE sets Oracle Database to check the policy against the value after an INSERT or UPDATE operation. The check applies only to the security relevant columns that are included in the policy definition. In other words, the INSERT or UPDATE operation will fail only if the security relevant column that is defined in the policy is added or updated in the INSERT or UPDATE statement.	FALSE
static_policy	If you set this value to TRUE, then Oracle Database assumes that the policy function for the static policy produces the same predicate string for anyone accessing the object, except for SYS or the privileged user who has the EXEMPT ACCESS POLICY privilege.	FALSE
policy_type	Default is NULL, which means policy_type is decided by the value of the static_policy parameter. Specifying any of these policy types overrides the value of static_policy.	NULL
long_predicate	Default is FALSE, which means the policy function can return a predicate with a length of up to 4000 bytes. TRUE means the predicate text string length can be up to 32K bytes. Policies existing before the availability of the long_predicate parameter retain a 32K limit.	FALSE
sec_relevant_cols_opt	If you specify this parameter, then transparent sensitive data protection inputs the sensitive column on which the protection is enabled to the sec_relevant_cols parameter of the DBMS_RLS.ADD_POLICY procedure. Allowed values are for sec_relevant_cols_opt are as follows: • NULL enables the filtering defined with sec_relevant_cols to take effect. • DBMS_RLS.ALL_ROWS displays all rows, but with sensitive column values, which are filtered by the sec_relevant_cols parameter, they display as NULL.	NULL

Related Topics

- Attaching a Policy to a Database Table, View, or Synonym
  The `DBMS_RLS` PL/SQL package can attach a policy to a table, view, or synonym.

15.12.3 Tutorial: Creating a TSDP Policy That Uses Virtual Private Database Protection

This tutorial demonstrates how to incorporate Oracle Virtual Private Database protection with a transparent sensitive data protection policy.

- Step 1: Create the hr_appuser User Account
  First, you must create a sample user account and then grant this user the appropriate privileges.
- Step 2: Identify the Sensitive Columns
  As the sample user `tsdp_admin`, you are ready to identify sensitive columns to protect.
- Step 3: Create an Oracle Virtual Private Database Function
  TSDP will associate the Oracle VPD policy function with the VPD policy that is automatically created when the TSDP policy is enabled.
- Step 4: Create and Enable a Transparent Sensitive Data Protection Policy
  After you have created the VPD policy function, you can associate it with a transparent sensitive data protection policy.
- Step 5: Test the Transparent Sensitive Data Protection Policy
  Now, you are ready to test the transparent sensitive data protection policy.
- Step 6: Remove the Components of This Tutorial
  If you no longer need the components of this tutorial, then you can remove them.

15.12.3.1 Step 1: Create the hr_appuser User Account

First, you must create a sample user account and then grant this user the appropriate privileges.

1. Log in to a PDB as user `SYS` with the `SYSDBA` administrative privilege.

   ```
   sqlplus sys@pdb_name as sysdba
   Enter password: password
   ```

   To find the available PDBs in a CDB, log in to the CDB root container and then query the `PDB_NAME` column of the `DBA_PDBS` data dictionary view. To check the current container, run the `show con_name` command.

2. Create the following user accounts:

   ```
   GRANT CREATE SESSION TO hr_appuser IDENTIFIED BY password;
   GRANT CREATE SESSION TO tsdp_admin IDENTIFIED BY password;
   ```

   Replace *password* with a password that is secure.

3. Grant user `tsdp_admin` the following privileges:

   ```
   GRANT CREATE PROCEDURE TO tsdp_admin;
   GRANT EXECUTE ON DBMS_TSDP_MANAGE TO tsdp_admin;
   GRANT EXECUTE ON DBMS_TSDP_PROTECT TO tsdp_admin;
   GRANT EXECUTE ON DBMS_RLS to tsdp_admin;
   ```

4. Connect as user `SCOTT`.

```
CONNECT SCOTT@pdb_name
Enter password: password
```

5. Grant the `hr_appuser` the `READ` object privilege for the `EMP` table.

```
GRANT READ ON EMP TO hr_appuser;
```

Related Topics

- Guidelines for Securing Passwords
  Oracle provides guidelines for securing passwords in a variety of situations.

15.12.3.2 Step 2: Identify the Sensitive Columns

As the sample user `tsdp_admin`, you are ready to identify sensitive columns to protect.

1. Connect as user `tsdp_admin`.

```
CONNECT tsdp_admin@pdb_name
Enter password: password
```

2. Create the `salary_type` sensitive type:

```
BEGIN
 DBMS_TSDP_MANAGE.ADD_SENSITIVE_TYPE (
  sensitive_type  => 'salary_type',
  user_comment    => 'Type for SCOTT.EMP column');
END;
/
```

3. Associate the `salary_type` sensitive type with the `SCOTT.EMP` table.

```
BEGIN
 DBMS_TSDP_MANAGE.ADD_SENSITIVE_COLUMN (
 schema_name        => 'SCOTT',
 table_name         => 'EMP',
 column_name        => 'SAL',
 sensitive_type     => 'salary_type',
 user_comment       => 'Sensitive column addition of SALARY_TYPE');
END;
/
```

15.12.3.3 Step 3: Create an Oracle Virtual Private Database Function

TSDP will associate the Oracle VPD policy function with the VPD policy that is automatically created when the TSDP policy is enabled.

- To create the VPD policy function, use the `CREATE OR REPLACE FUNCTION` procedure, as follows:

```
CREATE OR REPLACE FUNCTION vpd_function (
  v_schema IN VARCHAR2,
  v_objname IN VARCHAR2)
 RETURN VARCHAR2 AS
BEGIN
 RETURN 'SYS_CONTEXT(''USERENV'',''SESSION_USER'') = ''HR_APPUSER''';
END vpd_function;
/
```

15.12.3.4 Step 4: Create and Enable a Transparent Sensitive Data Protection Policy

After you have created the VPD policy function, you can associate it with a transparent sensitive data protection policy.

1. Create the Transparent Sensitive Data Protection policy.

```
DECLARE
 vpd_feature_options DBMS_TSDP_PROTECT.FEATURE_OPTIONS;
 policy_conditions DBMS_TSDP_PROTECT.POLICY_CONDITIONS;
BEGIN
 vpd_feature_options ('policy_function') := 'vpd_function';
 vpd_feature_options ('sec_relevant_cols_opt') := 'DBMS_RLS.ALL_ROWS';
 dbms_tsdp_protect.add_policy('tsdp_vpd', DBMS_TSDP_PROTECT.VPD,
vpd_feature_options, policy_conditions);
END;
/
```

   In this example, the `vpd_feature_options` parameter refers to the `sec_relevant_cols_opt` parameter from the `DBMS_RLS.ADD_POLICY` procedure. When the TSDP policy is enabled, the VPD policy that is automatically created will have its `sec_relevant_cols` parameter (of `DBMS_RLS.ADD_POLICY`) set to the name of the sensitive column on which TSDP enables the VPD policy. If you had not used the `sec_relevant_cols_opt` parameter, then TSDP would not have used the `DBMS_RLS.ADD_POLICY sec_relevant_cols_opt` parameter.

2. Associate the `tsdp_vpd1` TSDP policy with the `salary_type` sensitive type.

```
BEGIN
 DBMS_TSDP_PROTECT.ASSOCIATE_POLICY(
 policy_name        => 'tsdp_vpd',
 sensitive_type     => 'salary_type',
 associate          => TRUE);
END;
/
```

3. Enable protection to enforce the Virtual Private Database policy on all columns identified as `SALARY_TYPE`:

```
BEGIN
 DBMS_TSDP_PROTECT.ENABLE_PROTECTION_TYPE(
  sensitive_type           => 'salary_type');
END;
/
```

15.12.3.5 Step 5: Test the Transparent Sensitive Data Protection Policy

Now, you are ready to test the transparent sensitive data protection policy.

1. Connect as user `hr_appuser`.

```
CONNECT hr_appuser@pdb_name
Enter password: password
```

2. Query the `SCOTT.EMP` table as follows:

```
SELECT SAL, COMM, EMPNO FROM SCOTT.EMP;
```

   The following output appears:

```
    SAL   COMM   EMPNO
--------- ------ ----------
    800          7369
   1600    300   7499
   1250    500   7521
   2975          7566
   1250   1400   7654
   2850          7698
```

```
   2450          7782
   3000          7788
   5000          7839
   1500     0    7844
   1100          7876
    950          7900
   3000          7902
   1300          7934
14 rows selected.
```

The `vpd_function` function enables user `hr_appuser` to see the salaries in the `SAL` column of the `EMP` table.

3. Connect as user `SCOTT` and then perform the same query.

```
CONNECT SCOTT@pdb_name
Enter password: password

SELECT SAL, COMM, EMPNO FROM SCOTT.EMP;
```

The following output appears:

```
    SAL   COMM   EMPNO
--------- ------ ----------
                   7369
           300   7499
           500   7521
                 7566
          1400   7654
                 7698
                 7782
                 7788
                 7839
             0   7844
                 7876
                 7900
                 7902
                 7934
14 rows selected.
```

Even though `SCOTT` owns the `EMP` table, the `vpd_function` function prevents him from seeing the salaries in the `SAL` column of this table

15.12.3.6 Step 6: Remove the Components of This Tutorial

If you no longer need the components of this tutorial, then you can remove them.

1. Connect as user `tsdp_admin`.

```
CONNECT tsdp_admin@pdb_name
Enter password: password
```

2. Run the following statements in the order shown.

```
BEGIN
 DBMS_TSDP_MANAGE.DROP_SENSITIVE_COLUMN (
   schema_name        => 'SCOTT',
   table_name         => 'EMP',
   column_name        => 'SAL');
END;
/
```

```
BEGIN
 DBMS_TSDP_MANAGE.DROP_SENSITIVE_TYPE(
 sensitive_type     => 'salary_type');
END;
/

BEGIN
 DBMS_TSDP_PROTECT.DROP_POLICY(
   policy_name     => 'tsdp_vpd');
END;
/
```

3. Connect as user `SYSTEM`.

```
CONNECT SYSTEM@pdb_name
Enter password: password
```

4. Drop the `tsdp_admin` and `hr_appuser` accounts.

```
DROP USER tsdp_admin CASCADE;
DROP USER hr_appuser
```

15.13 Using Transparent Sensitive Data Protection Policies with Unified Auditing

The transparent sensitive data protection and unified auditing procedures can combine the protections of these two features.

- About Using TSDP Policies with Unified Audit Policies
  You can configure transparent sensitive data protection policies to audit object actions using unified auditing.
- Unified Audit Policy Settings That Are Used with TSDP Policies
  Audit policy settings can be used in the `POLICY_ENABLE_OPTIONS` parameter for the `DBMS_TSDP_PROTECT.ADD_POLICY` or `DBMS_TSDP_PROTECT.ALTER_POLICY` procedure.

15.13.1 About Using TSDP Policies with Unified Audit Policies

You can configure transparent sensitive data protection policies to audit object actions using unified auditing.

The `DBMS_TSDP_PROTECT.ADD_POLICY` and `DBMS_TSDP_PROTECT.ALTER_POLICY` procedures enable you to specify settings from the `CREATE AUDIT POLICY`, `ALTER AUDIT POLICY`, `AUDIT POLICY`, and `COMMENT` SQL statements. The TSDP policy enables the creation of action audit-options for object-specific options in the policy, such as `INSERT` or `DELETE` operations. System-wide audit options are not supported. Therefore, the audited object type is always `TABLE`. Only standard actions (such as `INSERT`) are permitted. Component actions, such as creating policies for Oracle Label Security or other Oracle Database features, are not supported.

This feature works as follows:

1. You create a TSDP policy with the necessary unified audit settings.

   The TSDP policy uses parameter settings from the `CREATE AUDIT POLICY`, `AUDIT POLICY`, and `COMMENT` statements.

2. You associate the TSDP policy with the necessary sensitive types by using the `DBMS_TSDP_PROTECT.ASSOCIATE_POLICY` procedure.

3. You then enable TSDP protection by using any of the `DBMS_TSDP_PROTECT.ENABLE_PROTECTION_*` procedures.
4. You enable the TSDP policy. As part of the TSDP policy enablement process, Oracle Database internally creates a unified audit policy and then enables it on the list of target users and roles that you specified in the `DBMS_TSDP_PROTECT.ADD_POLICY` procedure.

   The name of the internal policy begins with `ORA$UNIFIED_AUDIT_` followed by a random alpha-numeric string (for example, `ORA$UNIFIED_AUDIT_6J6L3RSJSN2VAN0XF`). You can find this policy by querying the `POLICY_NAME` column of the `AUDIT_UNIFIED_POLICIES` data dictionary view. To find the names of the users and roles on which this internally created TSDP unified audit policy is enforced, query the `AUDIT_UNIFIED_ENABLED_POLICIES` view.
5. When users try to perform an action on the table that is being protected by the TSDP policy, then based on the TSDP unified audit policy configuration, a unified audit record is written to the unified audit trail for this object access. You can then query the `UNIFIED_AUDIT_TRAIL` view to see the unified audit record that was created because of the TSDP unified audit policy enforcement.
6. These protections remain in place until you disable the TSDP policy for this column. At that point, Oracle Database automatically disables and then drops the internal policy, because it is no longer necessary. (A unified audit policy must be disabled before it can be dropped.) If you re-enable the TSDP policy, then the internal policy is recreated.

Related Topics

- Unified Audit Policy Settings That Are Used with TSDP Policies
  Audit policy settings can be used in the `POLICY_ENABLE_OPTIONS` parameter for the `DBMS_TSDP_PROTECT.ADD_POLICY` or `DBMS_TSDP_PROTECT.ALTER_POLICY` procedure.

15.13.2 Unified Audit Policy Settings That Are Used with TSDP Policies

Audit policy settings can be used in the `POLICY_ENABLE_OPTIONS` parameter for the `DBMS_TSDP_PROTECT.ADD_POLICY` or `DBMS_TSDP_PROTECT.ALTER_POLICY` procedure.

These audit policy settings are from the `AUDIT`, `CREATE AUDIT POLICY`, and `ALTER AUDIT POLICY` statements.

The following table describes these settings.

Table 15-2 Unified Audit Policy Settings Used for TSDP Policies

Parameter	Description	Default
`ACTION_AUDIT_OPTIONS`	A string containing a comma-separated list of SQL actions. Valid actions are: `ALTER`, `AUDIT`, `COMMENT`, `DELETE`, `FLASHBACK`, `GRANT`, `INDEX`, `INSERT`, `LOCK`, `RENAME`, `SELECT`, `UPDATE` To configure the policy to audit all of these actions, specify the keyword `ALL`.	`ALL`

Table 15-2 (Cont.) Unified Audit Policy Settings Used for TSDP Policies

Parameter	Description	Default
AUDIT_CONDITION	SYS_CONTEXT (*namespace*, *attribute*) *operation* *value-list* In this syntax, *operation* can be any of the following operators: IN,| NOT IN, =, <, >, or <> If the audit condition contains a single quotation mark, then specify two single quotation marks instead of one, and enclose the SYS_CONTEXT in single quotations. For example: 'SYS_CONTEXT(''USERENV'', ''CLIENT_IDENTIFIER'') = ''myclient''''	NULL
EVALUATE_PER	Can be one of the following: • STATEMENT • SESSION • INSTANCE	STATEMENT
ENTITY_NAME	A string that contains a comma-separated list of users or roles. If you omit this parameter, then the audit policy is enabled for all users.	NULL (that is, all database users)
ENABLE_OPTION	Applies only if the ENTITY_NAME parameter is used. It specifies if the ENTITY_NAME is a BY user list, an EXCEPT user list, or a BY USERS WITH GRANTED ROLES role list. Valid settings are: • BY • EXCEPT • BY USERS WITH GRANTED ROLES	BY
UNIFIED_AUDIT_POLICY_COMMENT	A string that describes the unified audit policy that will be created	NULL

15.14 Using Transparent Sensitive Data Protection Policies with Fine-Grained Auditing

The transparent sensitive data protection and fine-grained auditing procedures can combine the protections of these two features.

- About Using TSDP Policies with Fine-Grained Auditing
  You can configure a Transparent Sensitive Data Protection policy for fine-grained auditing.
- Fine-Grained Auditing Parameters That Are Used with TSDP Policies
  DBMS_FGA.ADD_POLICY settings can be used in the POLICY_ENABLE_OPTIONS parameter for the DBMS_TSDP_PROTECT.ADD_POLICY or DBMS_TSDP_PROTECT.ALTER_POLICY procedure.

15.14.1 About Using TSDP Policies with Fine-Grained Auditing

You can configure a Transparent Sensitive Data Protection policy for fine-grained auditing.

The DBMS_TSDP_PROTECT.ADD_POLICY and DBMS_TSDP_PROTECT.ALTER_POLICY procedures enable you to specify settings from the DBMS_FGA.ADD_POLICY procedure.

This feature works as follows:

1. You create a TSDP policy with the necessary fine-grained audit settings.

   The TSDP policy uses parameter settings from the `DBMS_FGA.ADD_POLICY` procedure.

2. You associate the TSDP policy with the necessary sensitive types by using the `DBMS_TSDP_PROTECT.ASSOCIATE_POLICY` procedure.

3. You then enable TSDP protection by using any of the `DBMS_TSDP_PROTECT.ENABLE_PROTECTION_*` procedures.

4. You enable the TSDP policy. As part of the TSDP policy enablement process, Oracle Database internally creates a fine-grained audit policy that you specified in the `DBMS_TSDP_PROTECT.ADD_POLICY` procedure.

   The name of the internal policy begins with `ORA$FGA_` followed by a random alpha-numeric string (for example, `ORA$FGA_6J6L3RSJSN2VAN0XF`). You can find this policy by querying the `POLICY_NAME` column of the `DBA_POLICIES` data dictionary view.

5. When users try to perform an action on the table that is being protected by the TSDP policies, then based on the policy configuration, a fine-grained audit record is generated in the `DBA_FGA_AUDIT_TRAIL` data dictionary view for this object access.

6. These protections remain in place until you disable the TSDP policy for this column. At that point, Oracle Database automatically drops the internal policy, because it is no longer needed. If you reenable the TSDP policy, then the internal policy is recreated.

Related Topics

- Fine-Grained Auditing Parameters That Are Used with TSDP Policies
  `DBMS_FGA.ADD_POLICY` settings can be used in the `POLICY_ENABLE_OPTIONS` parameter for the `DBMS_TSDP_PROTECT.ADD_POLICY` or `DBMS_TSDP_PROTECT.ALTER_POLICY` procedure.

15.14.2 Fine-Grained Auditing Parameters That Are Used with TSDP Policies

`DBMS_FGA.ADD_POLICY` settings can be used in the `POLICY_ENABLE_OPTIONS` parameter for the `DBMS_TSDP_PROTECT.ADD_POLICY` or `DBMS_TSDP_PROTECT.ALTER_POLICY` procedure.

The following table describes these settings.

Table 15-3 Fine-Grained Audit Policy Settings Used for TSDP Policies

Parameter	Description	Default
`audit_condition`	Specifies a Boolean value to indicate a monitoring condition, using the following syntax: *`operator value`* For example: `< 1000`	`NULL`
`handler_schema`	Schema that contains the event handler. The default, `NULL`, enables the current schema to be used.	`NULL`

Table 15-3 (Cont.) Fine-Grained Audit Policy Settings Used for TSDP Policies

Parameter	Description	Default
handler_module	Function name of the event handler. Include 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, then the user's SQL statement fails as well.	NULL
statement_types	You can specify one of the following statement types: INSERT, UPDATE, SELECT, or DELETE.	SELECT
audit_trail	If you have not yet migrated the database to full unified auditing, then use this setting to set the destination of the audit records: DB for the database or XML for XML records. This setting also specifies whether to populate the LSQLTEXT and LSQLBIND columns in the FGA_LOG$ system table. If full unified auditing is enabled, then Oracle Database ignores this parameter and writes the audit records to the unified audit trail.	NULL
object_schema	The schema that corresponds to the sensitive column	Schema that contains the sensitive column
object_name	The table that contains the sensitive column	The object (table or view) that contains the sensitive column
policy_name	A system-generated name for the internal fine-grained audit policy	Internal fine-grained audit policy system-generated name
audit_column	The sensitive column	The sensitive column
audit_column_opts	Determines whether to audit all or specific columns	DBMS_FGA.ANY_COLUMN
enable	Enable status for the TSDP policy; can be either TRUE or FALSE	TRUE
policy_owner	User who invokes the DBMS_TSDP_PROTECT.ENABLE_PROTECTION_* procedure	Current user

15.15 Using Transparent Sensitive Data Protection Policies with TDE Column Encryption

The TSDP procedures and Transparent Data Encryption column encryption statements can combine the protections of these two features.

- About Using TSDP Policies with TDE Column Encryption
  A TSDP policy can enable the encryption of columns that use Transparent Data Encryption.

- TDE Column Encryption ENCRYPT Clause Settings Used with TSDP Policies
  The `CREATE TABLE` and `ALTER TABLE` statement `ENCRYPT` clause settings can be used in the `POLICY_ENABLE_OPTIONS` parameter for the `DBMS_TSDP_PROTECT.ADD_POLICY` or `DBMS_TSDP_PROTECT.ALTER_POLICY` procedure.

15.15.1 About Using TSDP Policies with TDE Column Encryption

A TSDP policy can enable the encryption of columns that use Transparent Data Encryption.

The `DBMS_TSDP_PROTECT.ADD_POLICY` and `DBMS_TSDP_PROTECT.ALTER_POLICY` procedures enable you to specify the `ENCRYPT` clause settings from the `CREATE TABLE` or `ALTER TABLE` statement.

This feature works as follows:

1. You can create a TSDP policy by using the `DBMS_TSDP_PROTECT.ADD_POLICY` procedure. In the `ADD_POLICY` procedure, you can configure the policy for column encryption by setting the `SECURITY_FEATURE` parameter to `DBMS_TSDP_PROTECT.COLUMN_ENCRYPTION`. This setting enables encryption on the sensitive column when the TSDP policy is enabled on the object.
2. You create a TSDP policy with the necessary table encryption settings.

   The TSDP policy uses TDE column encryption `ENCRYPT` clause parameter settings from the `CREATE TABLE` or `ALTER TABLE` SQL statement.
3. You associate the TSDP policy with the necessary sensitive types by using the `DBMS_TSDP_PROTECT.ASSOCIATE_POLICY` procedure.
4. You then enable TSDP protection by using any of the `DBMS_TSDP_PROTECT.ENABLE_PROTECTION_*` procedures.
5. You enable the TSDP policy. At this point, Oracle Database creates an internal TSDP policy that uses the encrypted table settings that you created earlier in this procedure.

   The name of the internal policy begins with `ORA$TDECE_` followed by a random alpha-numeric string (for example, `ORA#TDECE_6J6L3RSJSN2VAN0XF`). You can find this policy by querying the `TSDP_POLICY` column of `DBA_TSDP_POLICY_PROTECTION` view.
6. When users try to perform an action on the table that is being protected by the policies, the output for the column is based on both the TDE column protections and the TSDP policy that are now in place. You can check if the column has been encrypted after you enabled the TSDP policy by querying the `ENCRYPTION_ALG` column of the `DBA_ENCRYPTED_COLUMNS` view.
7. These protections remain in place until you disable the TSDP policy for this column. At that point, Oracle Database internally issues an `ALTER TABLE` statement on the table that contains the sensitive column, so that the sensitive column is decrypted. If you reenable the TSDP policy, then TSDP internally executes the `ALTER TABLE` statement with the `ENCRYPT` clause for the column.

> **Note:**
>
> It is possible to create two policies on the same column with each policy specifying a different encryption algorithm. In this case, the stronger of the two algorithms is enforced on the sensitive column.

Related Topics

- TDE Column Encryption ENCRYPT Clause Settings Used with TSDP Policies
  The `CREATE TABLE` and `ALTER TABLE` statement `ENCRYPT` clause settings can be used in the `POLICY_ENABLE_OPTIONS` parameter for the `DBMS_TSDP_PROTECT.ADD_POLICY` or `DBMS_TSDP_PROTECT.ALTER_POLICY` procedure.

15.15.2 TDE Column Encryption ENCRYPT Clause Settings Used with TSDP Policies

The `CREATE TABLE` and `ALTER TABLE` statement `ENCRYPT` clause settings can be used in the `POLICY_ENABLE_OPTIONS` parameter for the `DBMS_TSDP_PROTECT.ADD_POLICY` or `DBMS_TSDP_PROTECT.ALTER_POLICY` procedure.

The following table describes these settings.

Table 15-4 TDE Column Encryption ENCRYPT Settings Used for TSDP Policies

Parameter	Description	Default
`encrypt_algorithm`	Available values • `3DES168` • `AES128` • `AES192` • `AES256` (default if none specified) • `ARIA128` • `ARIA192` • `ARIA256`	`AES256`
`salt`	Available values: • `SALT` • `NO SALT`	`SALT`
`integrity_algorithm`	Available values: • `SHA-1` • `NOMAC`	`SHA-1`

> **Note:**
>
> Starting with Oracle Database 23ai, the Transparent Data Encryption (TDE) decryption libraries for the GOST and SEED algorithms are deprecated, and encryption to GOST and SEED are desupported. Starting with Oracle Database 23ai, the Transparent Data Encryption (TDE) encryption libraries for the GOST and SEED algorithms are desupported and removed. The GOST and SEED decryption libraries are deprecated. Both are removed on HP Itanium platforms.
> GOST 28147-89 has been deprecated by the Russian government, and SEED has been deprecated by the South Korean government. If you need South Korean government-approved TDE cryptography, then use ARIA instead. If you are using GOST 28147-89, then you must decrypt and encrypt with another supported TDE algorithm. The decryption algorithms for GOST 28147-89 and SEED are included with Oracle Database 23ai, but are deprecated, and the GOST encryption algorithm is desupported with Oracle Database 23ai. If you are using GOST or SEED for TDE encryption, then Oracle recommends that you perform an online rekey operation before upgrading to Oracle Database 23ai. However, with the exception of the HP Itanium platform, the GOST and SEED decryption libraries are available with Oracle Database 23ai, so you can also decrypt after upgrading.

15.16 Transparent Sensitive Data Protection Data Dictionary Views

Oracle Database provides data dictionary views that list information about transparent sensitive data protection policies.

Table 15-5 describes these views. Before you can use these views, you must be granted the `SELECT_CATALOG_ROLE` role.

Table 15-5 Transparent Sensitive Data Protection Views

View	Description
`DBA_DISCOVERY_SOURCE`	Describes discovery import information with regard to transparent sensitive data protection policies
`DBA_SENSITIVE_COLUMN_TYPES`	Describes the sensitive column types that have been defined for the current database
`DBA_SENSITIVE_DATA`	Describes the sensitive columns in the database
`DBA_TSDP_IMPORT_ERRORS`	Shows information regarding the errors encountered during import of discovery result. It shows information with regard to the error code, schema name, table name, column name, and sensitive type.
`DBA_TSDP_POLICY_CONDITION`	Describes the transparent sensitive data protection policy and condition mapping. This view also lists the property-value pairs for the condition.
`DBA_TSDP_POLICY_FEATURE`	Shows the transparent sensitive data protection policy security feature mapping. (At this time, only Oracle Data Redaction and Oracle Virtual Private Database are supported.)
`DBA_TSDP_POLICY_PARAMETER`	Describes the parameters of transparent sensitive data protection policies

Table 15-5 (Cont.) Transparent Sensitive Data Protection Views

View	Description
DBA_TSDP_POLICY_PROTECTION	Shows the list of columns that have been protected through transparent sensitive data protection
DBA_TSDP_POLICY_TYPE	Shows the policy to sensitive column type mapping

Related Topics

- *Oracle Database Reference*