Oracle® Database
Data Redaction Guide

23ai

G12882-03

April 2025

Oracle Database Data Redaction Guide, 23ai

G12882-03

Contents

Preface

Welcome to the *Oracle Database Data Redaction Guide*. This guide describes how to implement, configure, and administer Data Redaction.

Audience

Oracle Database Data Redaction Guide is intended for users and systems professionals involved with the implementation, configuration, and administration of Data Redaction including:

- Implementation consultants
- System administrators
- Security administrators
- Database administrators (DBAs)

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customer access to and use of Oracle support services will be pursuant to the terms and conditions specified in their Oracle order for the applicable services.

Diversity and Inclusion

Oracle is fully committed to diversity and inclusion. Oracle respects and values having a diverse workforce that increases thought leadership and innovation. As part of our initiative to build a more inclusive culture that positively impacts our employees, customers, and partners, we are working to remove insensitive terms from our products and documentation. We are also mindful of the necessity to maintain compatibility with our customers' existing technologies and the need to ensure continuity of service as Oracle's offerings and industry standards evolve. Because of these technical constraints, our effort to remove insensitive terms is ongoing and will take time and external cooperation.

Related Documents

Before you configure Oracle Advanced Security features, you should be familiar with the following guides:

- *Oracle Database Administrator’s Guide*
- *Oracle Database Security Guide*

- *Oracle Database SQL Language Reference*
- *Oracle Database Reference*
- *Oracle Database PL/SQL Packages and Types Reference*
- *Oracle Multitenant Administrator's Guide*

Many books in the documentation set use the sample schemas of the default database. Refer to *Oracle Database Sample Schemas* for information about how these schemas were created and how you can use them.

Oracle Technical Services

To download the product data sheet, frequently asked questions, links to the latest product documentation, product download, and other collateral, visit Oracle Technical Resources (formerly Oracle Technology Network). You must register online before using Oracle Technical Services. Registration is free and can be done at

`https://www.oracle.com/technical-resources/`

My Oracle Support

You can find information about security patches, certifications, and the support knowledge base by visiting My Oracle Support (formerly Oracle*MetaLink*) at

`https://support.oracle.com`

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

Changes in This Release for *Oracle Database Data Redaction Guide*

This preface contains:

Enhancements to Oracle Data Redaction

This release includes many enhancements to Oracle Data Redaction such as, the optimization of existing capabilities and the removal of previous limitations.

Complex Expression on Redacted Columns in Views

In this release, SQL expressions such as, `CONCAT`, `SUM`, `TRIM`, `MIN`, or `MAX`, that involve a redacted column can be used in the `SELECT` list of a `CREATE VIEW` definition or an inline view.

In the following examples, `c1` has a redaction policy defined.

Example 1   Create View Statement with SQL Expression

```
create view v1 as select sum(c1) sum_c1 from t1 order by c1;
select * from v1;
```

Example 2   Inline View Support

```
select * from (select sum(c1) sum_c1 from t1 order by c1);
```

Extended Statistics and Functional Indexes on Redacted Columns

Oracle Data Redaction supports redaction policies on the base columns of function-based indexes and extended statistics beginning with Oracle Database 23ai.

This allows customers to combine these database performance features with database security capabilities. For more information on function-based indexes and extended statistics see the *Oracle Database Development Guide* and the *Oracle Database SQL Tuning Guide*.

GROUP BY on Expression On Columns With Redaction Policies

In this release, SQL expression on columns with redaction policy can be used in the `SELECT` list and `GROUP BY` clause.

A `GROUP BY` clause allows you to group the selected rows based on the value for each row and return a single row of summary information for each group. In the following example, `c1` has a redaction policy defined on it.

Example 3 `GROUP BY` With Redaction Policy on `c1`

```
select (c1+5)
from t1
group by (c1+5);
```

DISTINCT with ORDER BY on Redacted Column

In this release, a `DISTINCT` clause with `ORDER BY` clause on columns containing a Data Redaction policy can be used.

A `DISTINCT` clause specifies you want the database to return only one copy of each set of duplicate rows selected.

A `ORDER BY` clause allows you to specify the order in which you want the selected rows returned.

In the following example, `c1` has a redaction policy defined on it.

Example 4 `DISTINCT` With Redaction Policy on `c1`

```
select distinct c1
from t1
order by c1;
```

Inline View with Set Operation Having a Subquery

This release supports columns that contain a Data Redaction policy used by queries that have an inline view with set operation having a subquery.

A subquery in the `FROM` clause of a `SELECT` statement is also called an inline view.

Set Operator with Redacted Columns

In this release, a column in a set operation can have a redaction policy with different values for its properties than the redaction policy in the corresponding column of the set operator.

The properties of redaction policy which can be different are:

- Function type
- Function parameters or `REGEXP` parameters
- Policy expression
- Enable flag

In addition, a set operation query can have a column with redaction policy on one side of the set operation with the corresponding column on the other side of the set operation having no redaction policy.

Set operators combine the rows returned by two `SELECT` statements into a single result using `UNION`, `UNION ALL`, `INTERSECT`, `INTERSECT ALL`, `EXCEPT`, `EXCEPT ALL`, `MINUS`, or `MINUS ALL`.

Example 5 `SELECT` Statements with Full Redaction on `c1` and Partial Redaction on `c2`

```
select c1 from t1
union
select c2 from t1
```

The result of this query will be fully redacted as described in Redaction Results for Queries with SQL Expressions and Set Operation When Using Columns With Redaction Policies.

Redaction Results for Queries with SQL Expressions and Set Operation When Using Columns With Redaction Policies

In this release, if a SQL expression on columns with redaction policy is present in the `SELECT` list of a query, the result of the SQL expression is redacted using full redaction regardless of the underlying redaction type.

Similarly, if columns with redaction policy are present on either side of a set operation, the result of the set operation is redacted using full redaction regardless of the underlying redaction type.

Related Topics

- Redaction Results for Queries with SQL Expressions and Set Operation When Using Columns With Redaction Policies
  If a SQL expression is present in the `SELECT` list of query involving columns with redaction policies, the result of the SQL expression will always be fully redacted. The redaction type specified by the user using the `function_type` parameter will not be used to redact the data.

Optimization of Policy Expressions

In this release, when policy expression `1=1` is used, it is not evaluated as it always evaluates to `TRUE`. This leads to better performance during execution of the `SELECT` queries using the columns with redaction policy having the policy expression `1=1`.

Schema Privileges to Simplify Access Control in Oracle Data Redaction

Oracle Data Redaction supports the schema privileges introduced in Oracle Database 23ai.

This enhancement is as follows:

- The `ADMINISTER REDACTION POLICY` privilege must be granted to users as either a system privilege or a schema privilege for using the `DBMS_REDACT` PL/SQL package and not `CREATE TABLE` or `CREATE ANY TABLE`, which was required in Oracle Database 21c. This privilege is required in addition to the `EXECUTE` privilege on the `DBMS_REDACT` package for creating and managing data redaction policies.
- The `EXEMPT REDACTION POLICY` privilege can be granted as either a system privilege or a schema privilege.

Related Topics

- Privileges for Managing Oracle Data Redaction Policies
  Because data redaction involves the protection of highly sensitive data, only trusted users should create Oracle Data Redaction policies.
- *Oracle Database Security Guide*

BOOLEAN Data Type Supported in Oracle Data Redaction

Oracle Data Redaction supports the `BOOLEAN` data type introduced in Oracle Database 23ai.

As part of this enhancement, the `DBMS_REDACT.UPDATE_FULL_REDACTION_VALUES` procedure has a new parameter, `boolean_val`, to support changing the default value of `FALSE` for full redaction.

Related Topics

- Oracle Built-in Data Types Redaction Capabilities
  Oracle Data Redaction handles the Oracle built-in data types depending on the type of Data Redaction policies that are used.
- Syntax for the DBMS_REDACT.UPDATE_FULL_REDACTION_VALUES Procedure
  The `DBMS_REDACT.UPDATE_FULL_REDACTION_VALUES` procedure accommodates the standard supported Oracle Database data types.

1

Introduction to Oracle Data Redaction

Oracle Data Redaction enables you to redact data that is returned from queries issued by applications.

1.1 What Is Oracle Data Redaction?

Oracle Data Redaction enables you to obscure data that is returned from queries without affecting the data directly. This is done by redacting (selectively removing or obscuring sensitive or confidential information) the results of database queries just before presenting the results.

You can apply a Data Redaction policy to a table, view, or materialized view. Then, you apply the policy on the table to specific columns in the table. You can redact column data by using one of the following methods:

- **Full redaction.** You redact all of the contents of the column data. The redacted value returned to the querying user depends on the data type of the column and what the full redaction value was set to. By default, columns of the `NUMBER` data type are redacted with a zero (`0`), and character data types are redacted with a single space. However, these values can be changed using `UPDATE_FULL_REDACTION_VALUES`.
- **Partial redaction.** You redact a portion of the column data. For example, you can redact a Social Security Number with asterisks (*), except for the last 4 digits.
- **Regular expressions.** You can use regular expressions to look for patterns of data to redact. For example, you can use regular expressions to redact email addresses, which can have varying character lengths. It is designed for use with character data only.
- **Random redaction.** The redacted data presented to the querying user appears as randomly generated values each time it is displayed, depending on the data type of the column.
- **Nullify redation.** The Nullify redaction type redacts all the data in a column and replaces it with null values.
- **No redaction.** The None redaction type option enables you to test the internal operation of your redaction policies, with no effect on the results of queries against tables with policies defined on them. You can use this option to test the redaction policy definitions before applying them to a production environment.

Data Redaction is transparent to end users because it preserves the original data type and (optionally) the formatting. It is transparent to the database because the data remains the same in buffers, caches, and storage—only being changed at the last minute just before SQL query results are returned to the caller. The redaction is enforced consistently across all of the queries that access the tables or views with redaction policies. You can specify which users should see only redacted data by checking user information that is passed into the database through the `SYS_CONTEXT` function; you can redact data based on attributes of the current database; and you can implement multiple logical conditions within a given redaction policy. In addition, Data Redaction is implemented in a way that minimizes performance overhead. These characteristics make Oracle Data Redaction particularly well suited for usage by a range of applications, analytics tools, reporting tools, and monitoring tools that share common production databases.

Oracle Database applies the redaction at runtime when users access the data (that is, at query-execution time). This solution works well in a production system. During the time that the data is being redacted, all of the data processing is performed normally, and the back-end referential integrity constraints are preserved.

Data redaction can help you comply with industry, government, and organization requirements by restricting access to cardholder data or personally identifiable information based on a need-to-know.

Related Topics

- *Oracle Database Security Guide*
- Altering the Default Full Data Redaction Value
  The `DBMS_REDACT.UPDATE_FULL_REDACTION_VALUES` procedure alters the default full data redaction value.

1.2 When to Use Oracle Data Redaction

Unlike data masking, which masks sensitive data by permanently replacing it with fictitious but realistic data, data redaction redacts data as it is received from the database, leaving the data in the database unchanged.

Data Redaction enables you to easily disguise the data using several different redaction styles.

Oracle Data Redaction is particularly suited for call center applications, ad hoc reporting, and analytic workloads where the data is only available to the end user in a read-only format. However, take care when using Oracle Data Redaction with applications that perform updates back to the database because redacted data can be written back to this database.

For example, you may wish to redact:

- U.S. Social Security Numbers to only display the last four digits.
- American Express credit card numbers to only the last five digits.
- Fully redact salary, marital status, or medical data from views.
- Only certain columns in an analytic query, where the end user needs to see summaries of data but not actual values.

1.3 Benefits of Using Oracle Data Redaction

Oracle Data Redaction provides several benefits when you use it to protect your data.

These benefits are as follows:

- Applies the redaction at runtime, with no impact on underlying data or storage requirements
- Runs with little or no performance impact
- Requires no application changes
- Enables you to specify and manage Data Redaction policies through PL/SQL APIs or using the Oracle Enterprise Manger user web interface
- Is available for both on-premises and cloud databases

1.4 Example Use Cases for Oracle Data Redaction

Oracle Data Redaction addresses common use case scenarios.

1.4.1 Oracle Data Redaction for Sensitive Data in Read-Only Static Pages

Oracle Data Redaction enables you to redact sensitive data in application screens that have read-only static pages, such as dashboards and reports.

You can define a redaction policy on sensitive columns so that the data is redacted before it is passed to the application. Because the pages are for static display only, and data is not posted back to the database, redacted data would not be written back to the database and potentially corrupt records.

1.4.2 Oracle Data Redaction for Preventing Data Exposure by Management Tools

Oracle Data Redaction prevents sensitive data from being exposed by data management tools, such as tools for loading and viewing data.

Many applications include tools that enable users to load and manage their data. An example could be a SaaS application that allows subscribers to bulk load and manage customer information. You can define a redaction policy on the sensitive data so that sensitive data is redacted when it is displayed to these users when they perform these administrative activities.

1.4.3 Oracle Data Redaction to Prevent Disclosure of Data from Offline Analytics

Oracle Data Redaction prevents the disclosure of sensitive data to users who perform offline analytics on production data.

Oracle Data Redaction can be used to prevent the exposure of sensitive information to users who perform analytics on data that is contained in a data warehouse. You can define a redaction policy on sensitive data so that it is redacted as it is retrieved from the database and displayed to the user of the analytics software.

1.4.4 Oracle Data Redaction with Ad Hoc Database Queries Considerations

You may encounter situations where it is convenient to redact sensitive data for ad hoc queries that are performed by database users.

For example, in the course of supporting a production application, a user may need to run ad hoc database queries to troubleshoot and fix an urgent problem with the application. Even though Oracle Data Redaction is not designed to prevent data exposure to database users who run ad hoc queries directly against the database, it can provide an additional layer to reduce the chances of accidental data exposure. Because such users may have rights to change data, alter the database schema, and circumvent the SQL query interface entirely, it is possible for a malicious user to bypass Data Redaction policies in certain circumstances.

Be aware that Data Redaction does not place any restriction on the `WHERE` clause of ad hoc SQL, so the `WHERE` clause can be used in an iterative fashion to infer the actual data even when there is a Data Redaction policy on the queried column and only the redacted value is displayed.

Remember that the Oracle Database security tools are designed to be used together to improve overall security. By deploying one or more of these tools as a complement to Oracle Data Redaction, you can securely increase your overall security posture.

Related Topics

- Oracle Data Redaction General Security Guidelines
  It is important to understand the general security guidelines for using Oracle Data Redaction.

1.5 Oracle Data Redaction in a Multitenant Environment

Oracle Data Redaction policies are created and applied at the PDB level, and only apply to objects in that PDB. You cannot create redaction policies in the CDB root.

You cannot create a Data Redaction policy for a multitenant container database (CDB); you can only create the Data Redaction policy at the PDB level. This is because the objects for which you create Data Redaction policies typically reside in a PDB.

Related Topics

- Exemption of Users from Oracle Data Redaction Policies
  You can exempt users from having Oracle Data Redaction policies applied to the data they access.

2

Oracle Data Redaction Features and Capabilities

Oracle Data Redaction provides a variety of ways to redact different types of data.

2.1 Getting Started with Oracle Data Redaction

You can create and enable Oracle Data Redaction policies by using the DBMS_REDACT package. You define redaction policies by specifying the schema, object, and column.

- The **schema** level specifies the exact schema where one or more columns of an object have to be redacted.
- The **object** level includes tables, views, and materialized views where Orace Data Redaction policies are applied. You may apply a maximum of one policy per object.
- The **column** level includes where redaction functions operate. You can define a redaction function on one column when you create a redaction policy. You can then edit the policy to redact additional columns in the object with different functions if needed.

Note:

You can provide the same policy name for different objects but the policies will be different and unrelated. Policies are unique based on a combination of the object owner, the object name, and the policy name. If you decide to reuse the same policy name on a different objects, it is a separate policy and will have separate policy expressions.

For example, you can create an Oracle Data Redaction policy on the SALARY column of an HR.EMPLOYEES table with the following procedure:

```
BEGIN
 DBMS_REDACT.ADD_POLICY(
   object_schema       => 'hr',
   object_name         => 'employees',
   column_name         => 'salary',
   policy_name         => 'hr_emp_redact_comp_pol',
   function_type       => DBMS_REDACT.FULL,
   expression          => '1=1');
END;
```

In this case, we use the ADD_POLICY procedure in the DBMS_REDACT package to define an Oracle Data Redaction policy called hr_emp_redact_comp_pol. The function type DBMS_REDACT.FULL specifies that full data redaction be performed in retrieved values in the salary column, which means that by default, number data types are replaced with zero (0) in the output text. The expression parameter sets the policy to perform the redaction if it evaluates to TRUE (1=1).

Oracle Data Redaction provides a variety of ways to redact different types of data, which are described in this section.

2.2 Full Data Redaction to Redact All Data

Full data redaction redacts the entire contents of a specified column of a table or view.

By default the output is displayed as follows:

- **Character data types:** The output is a single space.
- **Number data types:** The output is a zero (`0`).
- **Date-time data types:** The output is set to the first day of January, 2001, which appears as `01-JAN-2001`.

When you run the `DBMS_REDACT.ADD_POLICY` procedure, to set the `function_type` parameter setting for full redaction, you enter the following setting:

```
function_type    => DBMS_REDACT.FULL
```

Full redaction is the default and is used whenever a Data Redaction policy specifies the column but omits the `function_type` parameter setting.

You can use the `DBMS_REDACT.UPDATE_FULL_REDACTION_VALUES` procedure to change the full redaction output to different values. You can find the current values by querying the `REDACTION_VALUES_FOR_TYPE_FULL` data dictionary view.

Related Topics

- Syntax for Creating a Full Redaction Policy
  The `DBMS_REDACT.ADD_POLICY` procedure enables you to create a full redaction policy.
- Altering the Default Full Data Redaction Value
  The `DBMS_REDACT.UPDATE_FULL_REDACTION_VALUES` procedure alters the default full data redaction value.
- *Oracle Database Reference*
- *Oracle Database PL/SQL Packages and Types Reference*

2.3 Partial Data Redaction to Redact Portions of Data

In partial data redaction, you redact portions of the displayed output.

You can set the position within the actual data at which to begin the redaction, the number of characters to redact starting from that position, and the redaction character to use. (Actual data is the data in a protected table or view. An example of actual data could be the number 123456789, and the redacted data version of this number could be 999996789.) This type of redaction is useful for situations where you want it to be obvious to the person viewing the data that it was redacted in some way. Typically, you use this type of redaction for credit card numbers or ID numbers.

Be aware that partial data redaction requires that your data width remain fixed. If you want to redact columns containing string values of variable length, then you must use regular expression redaction.

To specify partial redaction, you must set the `DBMS_REDACT.ADD_POLICY` or `DBMS_REDACT.ALTER_POLICY` procedure `function_type` parameter to `DBMS_REDACT.PARTIAL` and use the `function_parameters` parameter to define the partial redaction behavior.

The displayed output for partial data redaction can be as follows:

- **Character data types:** When partially redacted, a Social Security Number (represented as a hyphenated string within a character data type) with value `987-65-4320` could be redacted so that it is displayed as shown in the following examples. The code on the right specifies how to redact the character data: it specifies the expected input format of the actual data, the format to use for the display of the redacted output, the character to use for the redaction, the start position at which to begin the redaction, and how many characters to redact. The first example uses a predefined format (in previous releases called a shortcut) for character data type Social Security Numbers, and the second example replaces the first five numbers with an asterisk (`*`) while preserving the hyphens (`-`) in between the numbers.
  - `function_parameters => DBMS_REDACT.REDACT_US_SSN_F5` results in `XXX-XX-4320`
  - `function_parameters => 'VVVFVVFVVVV,VVV-VV-VVVV,*,1,5'` result in `***-**-4320`

  In these examples, `V` describes each character that potentially can be redacted, and `F` describes each character that you want to format using a formatting character.
- **Number data types:** Partially redacted `NUMBER` data types appear with some numerals replaced with specified characters. For example, a Social Security Number stored as `987654321` could appear as follows. Both redact the first five digits. The first example uses a predefined format that is designed for Social Security Numbers in the `NUMBER` data type, and the second replaces the first five numbers with the number `9`, starting from the first digit.
  - `function_parameters => DBMS_REDACT.REDACT_NUM_US_SSN_F5` results in `XXXXX4321`
  - `function_parameters => '9,1,5'` results in `999994321`
- **Date-time data types:** Partially redacted datetime values can appear simply as different dates. For example, the date `29-AUG-11 10.20.50.000000 AM` could appear as follows. In the first example, the day of the month is redacted to `02` (using the setting `d02`) and in the second example, the month is redacted to `DEC` (using `m12`). The uppercase values show the actual day (`D`), actual month (`M`), actual year (`Y`), actual hour (`H`), actual minute (`M`), and actual second (`S`).
  - `function_parameters => 'Md02YHMS'` results in `02-AUG-11 10.20.50.000000 AM`
  - `function_parameters => 'm12DYHMS'` results in `29-DEC-11 10.20.50.000000 AM`

Related Topics

- Regular Expressions to Redact Patterns of Data
  Regular expressions redact specific data within a column data value, based on a pattern search. You can use regular expressions to redact a column of strings of different lengths.
- Syntax for Creating a Partial Redaction Policy
  The `DBMS_REDACT.ADD_POLICY` statement enables you to create policies that redact specific parts of the data returned to the application.

2.4 Regular Expressions to Redact Patterns of Data

Regular expressions redact specific data within a column data value, based on a pattern search. You can use regular expressions to redact a column of strings of different lengths.

For example, you can redact the user name of email addresses, so that only the domain shows (for example, replacing `hpreston` in the email address `hpreston@example.com` with `[redacted]` so that it appears as `[redacted]@example.com`). To perform the redaction, set the

`DBMS_REDACT.ADD_POLICY` or `DBMS_REDACT.ALTER_POLICY` procedure `function_type` parameter to either `DBMS_REDACT.REGEXP` or `DBMS_REDACT.REGEXP_WIDTH`, and then use the following parameters to build the regular expression:

- A string search pattern (that is, the values to search for), such as:

  ```
  regexp_pattern => '(.+)@(.+\.[A-Za-z]{2,4})'
  ```

  This setting looks for a pattern of the following form:

  `one_or_more_characters@one_or_more_characters.2-4_characters_in_range_A-Z_or_a-z`

- A replacement string that replaces the value matched by the `regexp_pattern` setting. The replacement string can include back references to sub-expressions of the main regular expression pattern. The following example replaces the data before the `@` symbol (from the `regexp_pattern` setting) with the text `[redacted]`. The `\2` setting refers to the second match group, which is `(.+\.[A-Za-z]{2,4})` from the `regexp_pattern` setting.

  ```
  regexp_replace_string => '[redacted]@\2'
  ```

- The starting position for the string search string, such as the first character of the data, such as:

  ```
  regexp_position => DBMS_REDACT.RE_BEGINNING
  ```

  This value is the default if it is not specified.

- The kind of search and replace operation to perform, such as the first occurrence, every fifth occurrence, or all of the occurrences, such as:

  ```
  regexp_occurrence => DBMS_REDACT.RE_ALL
  ```

  This value is the default if it is not specified.

- The default matching behavior for the search and replace operation, such as whether the search is case-sensitive (`i` sets it to be not case-sensitive):

  ```
  regexp_match_parameter => 'i'
  ```

In addition to the default parameters, you can use a set of predefined formats that enable you to use commonly used regular expressions for telephone numbers, email addresses, and credit card numbers.

Related Topics

- Syntax for Creating a Regular Expression-Based Redaction Policy
  The `regexp_*` parameters of the `DBMS_REDACT.ADD_POLICY` procedure can create a regular expression-based redaction policy.
- Regular Expression-Based Redaction Policies Using Formats
  The `DBMS_REDACT.ADD_POLICY` procedure supports both `regexp_pattern` and `regexp_replace_string` parameters.

2.5 Random Data Redaction to Generate Random Values

In random data redaction, the entire value is redacted by replacing it with a random value.

The redacted values displayed in the result set of the query change randomly each time application users run the query.

This type of redaction is useful in cases where you do not want it to be obvious that the data was redacted. It works especially well for number and date-time data types, where it is difficult to distinguish between random and real data.

The displayed output for random values changes based on the data type of the redacted column, as follows:

- **Character data types:** The random output is a mixture of characters (for example, `HTU[G{\pjkEWcK`). It behaves differently for the `CHAR` and `VARCHAR2` data types, as follows:
  - **CHAR data type:** The redacted output is always in the same character set as the character set of the column. The byte length of the redacted output is always the same as the column definition length (that is, the column length that was provided at the time of table creation). For example, if the column is `CHAR(20)`, then a string of 20 random characters is provided in the redacted output of the user's query.
  - **VARCHAR2 data type:** For random redaction of a `VARCHAR` data type, the redacted output is always in the same character set as the character set of the column. The length of the redacted output is limited based on the length of the actual data in the column. No characters in excess of the length of the actual data are displayed. For example, if the column is `VARCHAR2(20)` and the row being redacted contains actual data with a length of 12, then a string of 12 random characters (not 20) is provided in the redacted output of the user's query for that row.
- **Number data types:** Each actual number value is redacted by replacing it with a random, non-negative number. This redaction results in random numbers that do not exceed the precision of the actual data. For example, the number `987654321` can be redacted by replacing it with any of the numbers `12345678`, `13579`, `0`, or `987654320`, but not by replacing it with any of the numbers `987654321`, `99987654321`, or `-1`. The number `-123` could be redacted by replacing it with the numbers `122`, `0`, or `83`, but not by replacing it with any of the numbers `123`, `1123`, or `-2`.

  The only exception to the above is when the actual value is an integer between -1 and 9. In this case, the actual data is redacted by replacing it with a random, non-negative integer modulo ten (10).
- **Date-time data types:** When values of the date data type are redacted using random Data Redaction, Oracle Database displays them with random dates that are always different from those of the actual data.

The setting for using random redaction is as follows:

```
function_type => DBMS_REDACT.RANDOM
```

Related Topics

- Syntax for Creating a Random Redaction Policy
  A random redaction policy presents the redacted data to the querying application user as randomly generated values, based on the column data type.

2.6 Redaction Using Null Values

You can create an Oracle Data Redaction policy that redacts column data by showing only null values in the query result.

This feature enables you to use the `DBMS_REDACT.NULLIFY` function to hide all of the sensitive data in a table or view column and replace it with null values. You can set this function by using the `function_type` parameter of the `DBMS_REDACT.ADD_POLICY` or `DBMS_REDACT.ALTER_POLICY` procedure.

For example:

```
function_type => DBMS_REDACT.NULLIFY
```

Related Topics

- Creating a Nullify Redaction Policy
  You can create Oracle Data Redaction policies that return null values for the displayed value of the table or view column.

2.7 No Redaction for Testing Purposes

You can create a Data Redaction policy that does not perform redaction. This enables you to include redaction policies in your applications during testing, and then apply those polices to your data as you move the application to production.

This is useful for cases in which you have a redacted base table, yet you want a specific application user to have a view that always shows the actual data. By default, a view created on a redacted table is redacted with the same policy used for the base table. However, you can define a separate redaction policy for the view. You can create a new view of the redacted table and then define a Data Redaction policy for this view. When the application queries the view, the policy defined on the view is applied. The policy still exists on the base table, but no redaction is performed when the application queries using the view as long as the `DBMS_REDACT.NONE` `function_type` setting was used to create a policy on the view.

2.8 Comparison of Full, Partial, Regexp, Random, and Nullify Redaction Based on Data Types

The full, partial, regular expression, random, and nullify data redaction styles affect the Oracle built-in, ANSI, user-defined, and Oracle supplied types in different ways.

2.8.1 Oracle Built-in Data Types Redaction Capabilities

Oracle Data Redaction handles the Oracle built-in data types depending on the type of Data Redaction policies that are used.

Table 2-1 describes the Oracle Data Redaction support for Oracle built-in data types.

Table 2-1 Redaction Support for Oracle Built-in Data Types

Column Data Type	Full Redaction	Partial Redaction	Regexp Redaction	Random Redaction	Nullify Redaction
`BFILE`	No	No	No	No	No
`BINARY_DOUBLE`	Yes	Yes	No	Yes	Yes
`BINARY_FLOAT`	Yes	Yes	No	Yes	Yes
`BLOB`	Yes	No	No	No	Yes
`BOOLEAN`	Yes	No	No	Yes	Yes
`CHAR`	Yes	Yes	Yes	Yes	Yes
`CLOB`	Yes	No	Yes	No	Yes
`DATE`	Yes	Yes	No	Yes	Yes
`FLOAT`	Yes	Yes	No	Yes	Yes
`FLOAT`	Yes	Yes	No	Yes	Yes

Table 2-1 (Cont.) Redaction Support for Oracle Built-in Data Types

Column Data Type	Full Redaction	Partial Redaction	Regexp Redaction	Random Redaction	Nullify Redaction
`INTERVAL DAY TO SECOND`	No	No	No	No	No
`INTERVAL YEAR TO MONTH`	No	No	No	No	No
`LONG RAW`	No	No	No	No	No
`NCHAR`	Yes	Yes	Yes	Yes	Yes
`NCLOB`	Yes	No	Yes	No	Yes
`NUMBER`	Yes	Yes	No	Yes	Yes
`NVARCHAR2`	Yes	Yes	Yes	Yes	Yes
`RAW`	No	No	No	No	No
`ROWID`	No	No	No	No	No
`TIMESTAMP`	Yes	Yes	No	Yes	Yes
`TIMESTAMP WITH LOCAL TIME ZONE`	Yes	Yes	No	Yes	Yes
`TIMESTAMP WITH TIME ZONE`	Yes	Yes	No	Yes	Yes
`UROWID`	No	No	No	No	No
`VARCHAR2` (Including long `VARCHAR2` such as `VARCHAR2(20000)`)	Yes	Yes	Yes	Yes	Yes

2.8.2 ANSI Data Types Redaction Capabilities

Oracle Data Redaction converts ANSI data types in specific ways, depending on the type of redaction that the Data Redaction policy has.

Table 2-2 compares how the full, partial, regular expression, random, and nullify redaction styles work for ANSI data types, with regard to how they are converted and their support status.

Table 2-2 Redaction Support for the ANSI Data Types

Data Type	How Converted	Full Redaction	Partial Redaction	Regexp Redaction	Random Redaction	Nullify Redaction
`CHARACTER(n)`, `CHAR(n)`	Converted to `CHAR(n)`	Yes	Yes	Yes	Yes	Yes

Table 2-2 (Cont.) Redaction Support for the ANSI Data Types

Data Type	How Converted	Full Redaction	Partial Redaction	Regexp Redaction	Random Redaction	Nullify Redaction
`CHARACTER VARYING(n)`, `CHAR VARYING(n)`	Converted to `VARCHAR2(`*n*`)`	Yes	Yes	Yes	Yes	Yes
`NATIONAL CHARACTER(n)`, `NATIONAL CHAR(n)`, `NCHAR(n)`	Converted to `NCHAR(`*n*`)`	Yes	Yes	Yes	Yes	Yes
`NATIONAL CHARACTER VARYING(n)`, `NATIONAL CHAR VARYING(n)`, `NCHAR VARYING(n)`	Converted to `NVARCHAR2(`*n*`)`	Yes	Yes	Yes	Yes	Yes
`NUMERIC`[(*p*, *s*)] `DECIMAL`[(*p*, *s*)]	Converted to `NUMBER(`*p*,*s*`)`	Yes	Yes	No	Yes	Yes
`INTEGER`, `INT`, `SMALLINT`	Converted to `NUMBER(38)`	Yes	Yes	No	Yes	Yes
`FLOAT`, `DOUBLE PRECISION`	Converted to `FLOAT(126)`	Yes	Yes	No	Yes	Yes
`REAL`	Converted to `FLOAT(63)`	Yes	Yes	No	Yes	Yes
`GRAPHIC`, `LONG VARGRAPHIC`, `VARGRAPHIC`, `TIME`	No conversion	No	No	No	No	No

2.8.3 Built-in and ANSI Data Types Full Redaction Capabilities

For full redaction, the default redacted value depends on whether the data type is Oracle built-in or ANSI.

Table 2-3 shows the default settings for both Oracle built-in and ANSI data type columns that use full redaction.

Table 2-3 Default Settings and Categories for Columns That Use Full Redaction

Data Type	Default Redacted Value	Data Type Category
`CHARACTER`	Single space (" ")	Oracle built-in
`CHARACTER(n)`, `CHAR(n)`	Single space (" ")	ANSI
`CHARACTER VARYING(n)`, `CHAR VARYING(n)`	Single space (" ")	ANSI

Table 2-3 (Cont.) Default Settings and Categories for Columns That Use Full Redaction

Data Type	Default Redacted Value	Data Type Category
`NATIONAL CHARACTER(n)`, `NATIONAL CHAR(n)`, `NCHAR(n)`	Single space (" ")	ANSI
`NATIONAL CHARACTER VARYING(n)`, `NATIONAL CHAR VARYING(n)`, `NCHAR VARYING(n)`	Single space (" ")	ANSI
`NUMBER`	Zero (`0`)	Oracle built-in
`NUMERIC`[(*p*, *s*)] `DECIMAL`[(*p*, *s*)]	Zero (`0`)	Oracle built-in
`INTEGER`, `INT`, `SMALLINT`	Zero (`0`)	ANSI
`FLOAT`, `DOUBLE PRECISION`	Zero (`0`)	ANSI
`REAL`	Zero (`0`)	ANSI
`DATE-TIME`	`01-01-01` or `01-01-01 01:00:00`	Oracle built-in
`BLOB`	Oracle's raw representation of `[redacted]`[1]	Oracle built-in
`CLOB`	`[redacted]`	Oracle built-in
`NCLOB`	`[redacted]`	Oracle built-in

1 If you have changed the character set, then you may need to invoke the `DBMS_REDACT.UPDATE_FULL_REDACTION_VALUES` procedure to set the value to the raw representation in the new character set, as follows:

```
DECLARE
 new_red_blob BLOB;
BEGIN
 DBMS_LOB.CREATETEMPORARY(new_red_blob, TRUE);
 DBMS_LOB.WRITE(new_red_blob, 10, 1, UTL_RAW.CAST_TO_RAW('[redacted]'));
 dbms_redact.update_full_redaction_values(
  blob_val      => new_red_blob);
DBMS_LOB.FREETEMPORARY(new_red_blob);
END;
/
```

After you run this procedure, restart the database.

See also Altering the Default Full Data Redaction Value for more information about using the `DBMS_REDACT.UPDATE_FULL_REDACTION_VALUES` procedure.

Related Topics

- ANSI, DB2, and SQL/DS Data Types in *Oracle Database SQL Language Reference*

2.8.4 User-Defined Data Types or Oracle Supplied Types Redaction Capabilities

Several data types are not supported by Oracle Data Redaction.

Oracle Data Redaction does not support user-defined data types or Oracle supplied types, such as anytype types, JSON types, VECTOR types, XML types, and Oracle Spatial types.

2.9 Central Management of Data Redaction Named Policy Expressions

You can create named policy expressions that can be used in the columns of multiple tables and views.

By having named policy expressions, you can centrally manage all of the policy expressions within a database.

When you modify the policy expression, the change is reflected in all table columns that use the expression. The named policy expression takes precedence over the `expression` setting in the Data Redaction policy. To create the named policy expression, you must use the `DBMS_REDACT.CREATE_POLICY_EXPRESSION` procedure, and to apply the policy expression to a column, you use the `DBMS_REDACT.APPLY_POLICY_EXPR_TO_COL` procedure. This feature provides flexibility to redact different columns in a table or view, based on different runtime conditions.

For example, consider a use case that involves a customer care application. A customer calls the customer care center to request a return on a recent purchase. A level 1 support representative of the call center must first verify the order ID, customer name, and customer address before initiating the return. During the process, there is no need for the level 1 support representative to view the customer’s credit card number. So, the credit card number column is redacted when the support representative queries the customer details in the call center application. When the return is initiated, a sales representative from the return department may need to view the credit card number to process the return. However, there is no need for the sales representative to view the expiration date of the credit card. So, when the sales representative queries the customer details in the same application, the credit card number is visible but the expiration date is redacted.

In this use case, different columns in the customer details table must be redacted in different ways, based on who the logged in user is. Oracle Data Redaction simplifies the implementation of this use case by using named policy expressions. This type of policy expression enables you to define and associate different policy expressions on different columns in the same table or view. Moreover, you can centrally manage named policy expressions within a database. Any updates that you make to a named policy expression are immediately propagated to all of the associated table or view columns.

Related Topics

- Creating and Managing Multiple Named Policy Expressions
  A named, centrally managed Oracle Data Redaction policy expression can be used in multiple redaction policies and applied to multiple tables or views.

3

Configuring Oracle Data Redaction Policies

An Oracle Data Redaction policy defines how to redact data in a column based on the table column type and the type of redaction you want to use.

3.1 About Oracle Data Redaction Policies

An Oracle Data Redaction policy defines the conditions in which redaction must occur.

A Data Redaction policy has the following characteristics:

- Oracle Data Redaction polices are applied to tables, views, or materialized views. You can apply only one policy on each object.
- The Data Redaction policy defines the following:
  - What to redact
  - How the redaction should occur
  - When the redaction takes place
- A Data Redaction policy can fully redact values, partially redact values, or randomly redact values. In addition, you can define a Data Redaction policy to not redact any data at all, for when you want to test your policies in a test environment.
- A Data Redaction policy can be defined with a policy expression which allows for different application users to be presented with either redacted data or actual data, based on whether the policy expression returns `TRUE` or `FALSE`. Redaction takes place when the boolean result of evaluating the policy expression is `TRUE`. For security reasons, the functions and operators that can be used in the policy expression are limited to `SYS_CONTEXT` and a few others. User-created functions are not allowed. Policy expressions can make use of the `SYS_SESSION_ROLES` namespace with the `SYS_CONTEXT` function to check for enabled roles.
- Different named Data Redaction policy expressions can be created and then applied individually for different columns within the same table or view.
- Oracle Database performs the redaction at execution time, just before the data is returned to the query.

Table 3-1 lists the procedures in the `DBMS_REDACT` package.

Table 3-1 DBMS_REDACT Procedures

Procedure	Description
`DBMS_REDACT.ADD_POLICY`	Adds a Data Redaction policy to a table or view
`DBMS_REDACT.ALTER_POLICY`	Modifies a Data Redaction policy
`DBMS_REDACT.APPLY_POLICY_EXPR_TO_COL`	Applies a named Data Redaction policy expression to a table or view column
`DBMS_REDACT.CREATE_POLICY_EXPRESSION`	Creates a named Data Redaction policy expression

Table 3-1 (Cont.) DBMS_REDACT Procedures

Procedure	Description
`DBMS_REDACT.DISABLE_POLICY`	Disables a Data Redaction policy
`DBMS_REDACT.DROP_POLICY`	Drops a Data Redaction policy
`DBMS_REDACT.DROP_POLICY_EXPRESSION`	Drops a named Data Redaction policy expression
`DBMS_REDACT.ENABLE_POLICY`	Enables a Data Redaction policy
`DBMS_REDACT.UPDATE_FULL_REDACTION_VALUES`	Globally updates the full redaction value for a given data type. You must restart the database instance before the updated values can be used.
`DBMS_REDACT.UPDATE_POLICY_EXPRESSION`	Updates a named Data Redaction policy expression

Related Topics

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

3.2 Privileges for Managing Oracle Data Redaction Policies

Because data redaction involves the protection of highly sensitive data, only trusted users should create Oracle Data Redaction policies.

You must have the `EXECUTE` privilege on the `DBMS_REDACT` package in order to run its subprograms. Procedures in the interface are run with privileges of the current user.

In addition to the `EXECUTE` privilege on the `DBMS_REDACT` package to run its subprograms, you must have the `ADMINISTER REDACTION POLICY` privilege at the schema or system level:

- Syntax of the `ADMINISTER REDACTION POLICY` privilege grant if the privilege is to apply to all non-`SYS` schemas across the database:

  ```
  GRANT ADMINISTER REDACTION POLICY TO grantee;
  ```

- Syntax of the `ADMINISTER REDACTION POLICY` privilege grant if the privilege is to be restricted to a specific schema:

  ```
  GRANT ADMINISTER REDACTION POLICY ON SCHEMA schema TO grantee;
  ```

You do not need any Oracle Data Redaction-specific privileges to access the underlying tables or views that will be protected by the policy. See Exemption of Users from Oracle Data Redaction Policies for Oracle Data Redaction-specific privileges to access actual data.

To find the privileges that a user has been granted, you can query the `DBA_SYS_PRIVS` data dictionary view. To find schema-specific privileges that a user has been granted, you can query any of the `*SCHEMA_PRIVS` data dictionary views.

Related Topics

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

3.3 General Syntax of the DBMS_REDACT.ADD_POLICY Procedure

To create a Data Redaction policy, you must use the `DBMS_REDACT.ADD_POLICY` procedure.

The complete syntax for the `DBMS_REDACT.ADD_POLICY` procedure is as follows:

```
DBMS_REDACT.ADD_POLICY (
 object_schema                IN VARCHAR2 := NULL,
 object_name                  IN VARCHAR2,
 policy_name                  IN VARCHAR2,
 policy_description           IN VARCHAR2 := NULL,
 column_name                  IN VARCHAR2 := NULL,
 column_description           IN VARCHAR2 := NULL,
 function_type                IN BINARY_INTEGER := DBMS_REDACT.FULL,
 function_parameters          IN VARCHAR2 := NULL,
 expression                   IN VARCHAR2,
 enable                       IN BOOLEAN := TRUE,
 regexp_pattern               IN VARCHAR2 := NULL,
 regexp_replace_string        IN VARCHAR2 := NULL,
 regexp_position              IN BINARY_INTEGER :=1,
 regexp_occurrence            IN BINARY_INTEGER :=0,
 regexp_match_parameter       IN VARCHAR2 := NULL);
```

In this specification:

Table 3-2 `DBMS_REDACT.ADD_POLICY` Parameters

Parameter	Description
`object_schema`	Specifies the schema of the object on which the Data Redaction policy will be applied. If you omit this setting (or enter `NULL`), then Oracle Database uses the current user's name. Be aware that the meaning of "current user" here can change, depending on where you invoke the `DBMS_REDACT.ADD_POLICY` procedure. For example, suppose user `mpike` grants user `fbrown` the `EXECUTE` privilege on a definer's rights PL/SQL package called `mpike.protect_data` in `mpike`'s schema. From within this package, `mpike` has coded a procedure called `protect_cust_data`, which invokes the `DBMS_REDACT.ADD_POLICY` procedure. User `mpike` has set the `object_schema` parameter to `NULL`. When `fbrown` invokes the `protect_cust_data` procedure in the `mpike.protect_data` package, Oracle Database attempts to define the Data Redaction policy around the object `cust_data` in the `mpike` schema, not the `cust_data` object in the schema that belongs to `fbrown`.
`object_name`	**Required** Specifies the name of the table or view to which the Data Redaction policy applies.
`policy_name`	**Required** Specifies the name of the policy to be created. Ensure that this name is unique in the database instance. You can find a list of existing Data Redaction policies by querying the `POLICY_NAME` column of the `REDACTION_POLICIES` data dictionary view.

Table 3-2 (Cont.) `DBMS_REDACT.ADD_POLICY` Parameters

Parameter	Description
`expression`	**Required** Specifies a Boolean SQL expression to determine when the policy is applied. Redaction takes place only if this policy expression evaluates to `TRUE`. See Using Expressions to Define Conditions for Data Redaction Policies. By default, a Data Redaction policy expression applies to all the columns that belong to the Data Redaction policy defined on that table or view. Alternatively, you can choose to create and associate a policy expression for individual columns to override the existing expression. These column level expressions are called named policy expressions. See Creating and Managing Multiple Named Policy Expressions.
`policy_description`	Specifies a brief description of the purpose of the policy.
`column_name`	Specifies the column whose data you want to redact. Note the following: • **You can apply the Data Redaction policy to multiple columns.** If you want to apply the Data Redaction policy to multiple columns, then after you use `DBMS_REDACT.ADD_POLICY` to create the policy, run the `DBMS_REDACT.ALTER_POLICY` procedure as many times as necessary to add each of the remaining required columns to the policy. See Altering an Oracle Data Redaction Policy. • **Only one policy can be defined on a table or view.** You can, however, create a new view on the table, and by defining a second redaction policy on this new view, you can choose to redact the columns in a different way when a query is issued against this new view. When deciding how to redact a given column, Oracle Database uses the policy of the earliest view in a view chain. For example, suppose you have a table `table_1` and two views `view_1` and `view_2`, and each of these has a Data Redaction policy as follows: – `table_1` has the Data Redaction policy `table_1_pol`. – `view_1` is created on `table_1`. `view_1` has the Data Redaction policy `view_1_pol`. – `view_2` is created on `view_1`. `view_2` has the Data Redaction policy `view_2_pol`. If `table_1` is queried, then policy `table_1_pol` is applied on the columns. When `view_1` is queried, then policy `view_1_pol` is applied. Similarly, if `view_2` is queried, columns in `view_2` are redacted according to `view_2_pol`. • **If you do not specify a column (for example, by entering NULL), then no columns are redacted by the policy.** This enables you to create your policies so that they are in place, and then later on, you can add the column specification when you are ready. If you omit a column name when setting `DBMS_REDACT.ADD_POLICY`, then only the `expression` parameter is required. • **Do not use a column that is currently used in an Oracle Virtual Private Database (VPD) row filtering condition.** In other words, the column should not be part of the VPD predicate generated by the VPD policy function. (See Oracle Data Redaction and Oracle Virtual Private Database for more information about using Data Redaction with VPD.)
`column_description`	Specifies a brief description of the purpose of the policy.

Table 3-2 (Cont.) `DBMS_REDACT.ADD_POLICY` Parameters

Parameter	Description
`function_type`	Specifies a function that sets the type of redaction. Can be specified only when `column_name` is specified. If you omit the `function_type` parameter, then the default redaction `function_type` setting is `DBMS_REDACT.FULL`. See the following sections for more information: • Syntax for Creating a Full Redaction Policy • Syntax for Creating a Partial Redaction Policy • Syntax for Creating a Regular Expression-Based Redaction Policy • Syntax for Creating a Random Redaction Policy • Syntax for Creating a Nullify Redaction Policy • Syntax for Creating a Policy with No Redaction
`function_parameters`	Specifies how the column redaction should appear for partial redaction. Required if `function_type` is `DBMS_REDACT.PARTIAL`. See Syntax for Creating a Partial Redaction Policy.
`enable`	When set to `TRUE` (the default), enables the policy upon creation. When set to `FALSE`, it creates the policy as a disabled policy. After you create the policy, you can disable or enable it. See the following sections: • Disabling an Oracle Data Redaction Policy • Enabling an Oracle Data Redaction Policy
`regexp_pattern`	Enable you to use regular expressions to redact data, either fully or partially. If the `regexp_pattern` does not match anything in the actual data, then full redaction will take place, so be careful when specifying the `regexp_pattern`. Ensure that all of the values in the column conform to the semantics of the regular expression you are using. See Syntax for Creating a Regular Expression-Based Redaction Policy for more information.
`regexp_replace_string`	Specifies how you want to replace the data to be redacted. This data can be of any of the data types `CHAR`, `VARCHAR2`, `NCHAR`, or `NVARCHAR2`. The `regexp_replace_string` can contain up to 500 back references to subexpressions in the form `\n`, where n is a number from one to nine. If you want to include a backslash (`\`) in the `regexp_replace_string` setting, then you must precede it with the escape character, which is also a backslash. For example, to literally replace the matched pattern with `\2` (rather than replace it with the second matched subexpression of the matched pattern), you enter `\\2` in the `regexp_replace_string setting`. For more information, see *Oracle Database SQL Language Reference*.
`regexp_position`	Specifies the starting position for the string search. The value that you enter must be a positive integer indicating the character of the `column_name` data where Oracle Database should begin the search. The default is 1 or the `DBMS_REDACT.RE_BEGINNING` format, meaning that Oracle Database begins the search at the first character of the `column_name` data.

Table 3-2 (Cont.) `DBMS_REDACT.ADD_POLICY` Parameters

Parameter	Description
`regexp_occurrence`	Specifies how to perform the search and replace operation. The value that you enter must be a non-negative integer indicating the occurrence of the replace operation. If you specify `0` or the `DBMS_REDACT.RE_ALL` format, then Oracle Database replaces all the occurrences of the match. If you specify the `DBMS_REDACT.RE_FIRST` format, then Oracle Database replaces the first occurrence of the match. If you specify a positive integer `n`, then Oracle Database replaces the nth occurrence of the match. If the occurrence is greater than 1, then the database searches for the second occurrence beginning with the first character following the first occurrence of pattern, and so forth.
`regexp_match_parameter`	Specifies a text literal that lets you change the default matching behavior of the function. The behavior of this parameter is the same for this function as for the `REGEXP_REPLACE` SQL function. To filter the search so that it is not case sensitive, specify the `RE_CASE_INSENSITIVE` format. See *Oracle Database SQL Language Reference* for detailed information.

To find information about redaction policies, you can query the `REDACTION_POLICIES` data dictionary view.

The first time you add a full redaction policy to a boolean column after the `COMPATIBLE` database initialization parameter is set to 23 or higher, you will see the `BOOLEAN_VALUE` column appear in the catalog view for `REDACTION_VALUES_FOR_TYPE_FULL` and the `BOOLCOL` column appear in the `SYS.RADM_FPTM$` data dictionary table.

3.4 Planning an Oracle Data Redaction Policy

Before you create a Oracle Data Redaction policy, you should plan the data redaction policy that best suits your site's needs.

1. Ensure that you have the appropriate privileges to create and manage Oracle Data Redaction policies.
   - To create redaction policies on objects in your own schema, you must have the `EXECUTE` privilege on the `DBMS_REDACT` PL/SQL package and the `ADMINISTER REDACTION POLICY` system or schema privilege.
   - To create redaction policies on objects in another user's schema, you must have the `EXECUTE` privilege on the `DBMS_REDACT` PL/SQL package and the `ADMINISTER REDACTION POLICY` system privilege.

   For more information see Privileges for Managing Oracle Data Redaction Policies.
2. Determine the data type of the table or view column that you want to redact.

   For more information see Oracle Built-in Data Types Redaction Capabilities.
3. Determine if the base object to which you want to add the Data Redaction policy has dependent objects. For more information see Oracle Data Redaction and Object Dependencies.

   Oracle recommends that you only add a Data Redaction policy to an object that has dependent objects during off-peak hours or during a scheduled downtime.

4. Decide on the type of redaction that you want to perform: full, random, partial, regular expressions, nullify, or none.

   For more information see Comparison of Full, Partial, Regexp, Random, and Nullify Redaction Based on Data Types or What Is Oracle Data Redaction?.

5. Decide which users to apply the Data Redaction policy to.

   For more information see Using Expressions to Define Conditions for Data Redaction Policies.

6. Based on this information, create the Data Redaction policy by using the `DBMS_REDACT.ADD_POLICY` procedure.

   For more information see General Syntax of the DBMS_REDACT.ADD_POLICY Procedure.

7. Configure the policy to have additional columns to be redacted.

   If you need additional columns to have different condition for redaction, you can create a Data Redaction named policy expression.
   For example, if SSN should be redacted for all users, it can have a condition of `1=1`.

   For more information see Creating and Managing Multiple Named Policy Expressions.

After you create the Data Redaction policy, it is automatically enabled and ready to redact data.

Related Topics

- Redacting Multiple Columns
  You can redact more than one column in a Data Redaction policy.

3.5 Using Expressions to Define Conditions for Data Redaction Policies

The `expression` parameter in the `DBMS_REDACT.ADD_POLICY` or `DBMS_REDACT.ALTER_POLICY` procedures sets the conditions under which those policies apply.

3.5.1 About Using Expressions in Data Redaction Policies

The `DBMS_REDACT.ADD_POLICY` and `DBMS_REDACT.ALTER_POLICY` `expression` parameter define a boolean expression that must evaluate to `TRUE` to enable redaction.

The expression that is defined in the `expression` parameter is the default expression for the Oracle Data Redaction policy. If you apply a named policy expression for the columns that will be redacted by the Data Redaction policy, then the named policy expression takes precedence over the expression defined in the Data Redaction policy. If `1=1` is specified for the expression parameter which is a policy expression that evaluates to `TRUE`, it enables the redaction to be performed so long as the querying user is not exempt from the redaction policy. Using this condition saves CPU time during the fetch operation from a redacted column. By avoiding the need to evaluate any policy expression, the performance of fetching from any column that is protected by a Data Redaction policy is improved.

You can create expressions that make use of other Oracle Database features. For example, you can create expressions that are based on a user’s environment (using the `SYS_CONTEXT` and `XS_SYS_CONTEXT` functions), character string functions, the Oracle Label Security label dominance functions, or Oracle Application Express functions.

Follow these guidelines when you write the expression:

- Use only the following operators: AND, OR, IN, NOT IN, =, !=, <>, <, >, >=, <=
- Because the expression must evaluate to TRUE for redaction, be careful when making comparisons with NULL. Remember that in SQL the value NULL is undefined, so comparisons with NULL tend to return FALSE.
- Do not use user-created functions in the expression parameter; this is not permitted.
- Remember that for user SYS and users who have the EXEMPT REDACTION POLICY system or schema privilege, all of the Data Redaction policies are bypassed, so the results of their queries are not redacted. When you exempt a user from a Data Redaction policy, you should be aware of how the exemption is affected by users with specific privileges (such as the DBA role) and by Oracle Data Pump.

Related Topics

- Exemption of Users from Oracle Data Redaction Policies
  You can exempt users from having Oracle Data Redaction policies applied to the data they access.
- Oracle Data Pump Security Model for Oracle Data Redaction
  The DATAPUMP_EXP_FULL_DATABASE role includes the powerful EXEMPT REDACTION POLICY system privilege.

3.5.2 Supported Functions for Data Redaction Expressions

You can create expressions that use functions to return specific types of data, such as SYS_CONTEXT namespaces.

3.5.2.1 Expressions Using Namespace Functions

You can use the SYS_CONTEXT and XS_SYS_CONTEXT namespace functions in Data Redaction expressions.

Table 3-3 Expressions Using Namespace Functions

Namespace Function	Description
SYS_CONTEXT	Returns the value associated with a namespace. The following namespace functions are valid: • USERENV (default namespace), which includes values such as SESSION_USER and CLIENT_IDENTIFIER. • SYS_SESSION_ROLES, which contains attributes for each role • XS$SESSION, which contains attributes for the user session. • User-defined namespaces, but these must exist in the DBA_CONTEXT catalog view before the policy expression is created.
XS_SYS_CONTEXT	Similar to SYS_CONTEXT but designed for an Oracle Real Application Security environment. XS_SYS_CONTEXT supports the same namespaces that SYS_CONTEXT supports.

Related Topics

- *Oracle Database SQL Language Reference*
- *Oracle Database Real Application Security Administrator's and Developer's Guide*

3.5.2.2 Expressions Using the SUBSTR Function

You can use the `SUBSTR` function, which returns portion (such as characters 1–3) of the character string specified, in Data Redaction expressions. The first parameter must be a constant string or a call to the `SYS_CONTEXT` function or the `XS_SYS_CONTEXT` function.

Table 3-4 Expressions Using SUBSTR String Functions

SUBSTR String Function	Description
`SUBSTR`	Returns a portion of the input `char` value, beginning at *`character position`*, *`substring_length`* characters long. `SUBSTR` calculates length using characters as defined by the input character set.
`SUBSTRB`	Returns the specified portion of the input value in bytes
`SUBSTRC`	Returns the specified portion of the input value in Unicode complete characters
`SUBSTR2`	Returns the specified portion of the input value in UCS2 code points
`SUBSTR4`	Returns the specified portion of the input value in UCS4 code points

Related Topics

- *Oracle Database SQL Language Reference*

3.5.2.3 Expressions Using Length of Character String Functions

You can use functions that return the length of character strings, in Data Redaction expressions.

Oracle Database also checks that the arguments to each of these operators is either a constant string or a call to the `SYS_CONTEXT` or `XS_SYS_CONTEXT` function.

Table 3-5 Expressions Using Character String Functions

Character String Function	Description
`LENGTH`	Returns the length of the input `char` value. `LENGTH` calculates length using characters as defined by the input character set.
`LENGTHB`	Returns the length of the input value in bytes
`LENGTHC`	Returns the length of the input value in Unicode complete characters
`LENGTH2`	Returns the length of the input value in UCS2 code points
`LENGTH4`	Returns the length of the input value in UCS4 code points

Related Topics

- *Oracle Database SQL Language Reference*

3.5.2.4 Expressions Using Oracle Application Express Functions

You can use Oracle Application Express functions in Data Redaction expressions.

Table 3-6 Oracle Application Express Functions

Oracle Application Express Function	Description
`V`	Returns the session state for an item. It is a wrapper for the `APEX_UTIL.GET_SESSION_STATE` function
`NV`	Returns the numeric value for a numeric item. It is a wrapper for the `APEX_UTIL.GET_NUMERIC_SESSION_STATE` function

Related Topics

- *Oracle APEX API Reference*

3.5.2.5 Expressions Using Oracle Label Security Functions

You can use Oracle Label Security functions with Data Redaction expressions.

Table 3-7 Oracle Label Security Functions

Oracle Label Security Function	Description
`LBACSYS.OLS_LABEL_DOMINATES`	Checks if the session label of an Oracle Label Security policy dominates or is equal to another OLS label
`OLS_DOMINATES`*, `DOMINATES`*, and `S_DOM`*	Checks if one OLS label is dominant to a second OLS label. `DOMINATES` and `S_DOM`, which were used in releases earlier than Oracle Database 12.1, were deprecated in that release. Use the `OLS_DOMINATES` or `OLS_DOM` function instead.
`OLS_DOM`*	Checks if one OLS label is dominant to a second OLS label
`DOM`*	Checks if one OLS label is dominant to a second OLS label
`OLS_STRICTLY_DOMINATES`*	Checks if one OLS label is dominant to a second OLS label and is not equal to it
`STRICTLY_DOMINATES`*	Checks if one OLS label is dominant to a second OLS label and is not equal to it
`SA_UTL.DOMINATES`*	Checks if one OLS label dominates a second OLS label or if the session label for a given OLS policy dominates an OLS label
`SA_UTL.CHECK_READ`	Checks if a user can read a policy-protected row
`SA_UTL.NUMERIC_LABEL`	Returns the current session OLS label
`CHAR_TO_LABEL`	Converts a character string to an OLS label tag
`SA_SESSION.LABEL`	Returns the label that is associated with the specified OLS policy

* Oracle Data Redaction checks that their parameters are either constants or calls to only one of the `SA_UTL.NUMERIC_LABEL`, `CHAR_TO_LABEL`, and `SA_SESSION.LABEL` functions, and that the arguments to those functions are constant.

Related Topics

- Using *Oracle Label Security Administrator’s Guide*

3.5.3 Applying the Redaction Policy Based on User Environment

You can apply a Data Redaction policy based on the user’s environment, such as the session user name or a client identifier.

- Use the `USERENV` namespace of the `SYS_CONTEXT` function in the `DBMS_REDACT.ADD_POLICY` `expression` parameter to apply the policy based on a user’s environment.

  For example, to apply the policy only to the session user name `psmith`:

  ```
  expression  => 'SYS_CONTEXT(''USERENV'',''SESSION_USER'') = ''PSMITH'''
  ```

Related Topics

- *Oracle Database SQL Language Reference*

3.5.4 Applying the Redaction Policy Based on Database Roles

You can apply a Data Redaction policy based on a database role.

Use the `SYS_SESSION_ROLES` namespace in the `SYS_CONTEXT` function to apply the policy based on a user role.

This namespace contains attributes for each role. The value of the attribute is `TRUE` if the specified role is enabled for the querying application user; the value is `FALSE` if the role is not enabled. If the expression is `TRUE`, redacted data is returned to the user. If the expression is `FALSE`, actual data is returned to the user.

For example, suppose you wanted only supervisors to be allowed to see the actual data. The following example shows how to use the `DBMS_REDACT.ADD_POLICY` `expression` parameter to set the policy to show the actual data to any application user who has the `supervisor` role enabled, but redact the data for all of the other application users.

```
expression  => 'SYS_CONTEXT(''SYS_SESSION_ROLES'',''SUPERVISOR'') = ''FALSE'''
```

You can specify multiple roles with the `expression` parameter. The following example checks for the presence of the `SUPERVISOR`, `CLERK`, and `TEMP_WORKER` roles. Actual data will be displayed to a user with the `SUPERVISOR` role. Users who have the `CLERK` or `TEMP_WORKER` roles will see redacted data.

```
expression => 'SYS_CONTEXT(''SYS_SESSION_ROLES'',''SUPERVISOR'') = ''FALSE''
OR SYS_CONTEXT(''SYS_SESSION_ROLES'',''CLERK'') = ''TRUE''
OR SYS_CONTEXT(''SYS_SESSION_ROLES'',''TEMP_WORKER'') = ''TRUE'''
```

All roles are disabled in any named PL/SQL block that runs with definer's rights. Procedures and functions are created with definer's rights by default. For the policy expression with `SYS_SESSION_ROLES` to take effect, create the procedure or function to be invoker's rights so that the enabled roles are effective. Alternatively, you can use code based access controls to grant a role to a program unit (function, package, or procedure), allowing the program unit to use the role in either definer's or invoker's rights. For more information, see Managing Security for Definer's Rights and Invoker's Rights in the *Oracle Database Security Guide*.

Related Topics

- Roles Used in Named Blocks with Definer's Rights
- Roles Used in Named Blocks with Invoker's Rights and Anonymous PL/SQL Blocks
- *Oracle Database SQL Language Reference*

3.5.5 Applying the Redaction Policy Based on Application Express Session States

You can apply a Data Redaction policy based on an Oracle Application Express (APEX) session state.

- Use either of the following public Application Express APIs in the `DBMS_REDACT.ADD_POLICY` `expression` parameter to apply the policy on an Oracle Application Express session state:
  - `V`, which is a synonym for the `APEX_UTIL.GET_SESSION_STATE` function
  - `NV`, which is a synonym for the `APEX_UTIL.GET_NUMERIC_SESSION_STATE` function

  If you want redaction to take place when the querying user is *not* within the context of an APEX application (when the query is issued from outside the APEX framework, for example directly through SQL*Plus), then use an `IS NULL` clause as shown in the following example.

  For example, to set the `DBMS_REDACT.ADD_POLICY` `expression` parameter when the query is issued from outside the APEX framework, use an `IS NULL` clause, as follows:

  ```
  expression => 'V(''APP_USER'') != ''mavis@example.com'' or V(''APP_USER'') is null'
  ```

  You can, for example, use these functions to redact data based on a job or a privilege role that is stored in a session state in an APEX application.

Related Topics

- *Oracle APEX API Reference*

3.5.6 Applying the Redaction Policy Based on Oracle Label Security Label Dominance

You can set a condition on which to apply a Data Redaction policy based on the dominance of Oracle Label Security labels.

- Use the public standalone function `OLS_LABEL_DOMINATES` to check the dominance of a session label.

  This function returns `1` (`TRUE`) if the session label of the specified `policy_name` value dominates or is equal to the label that is specified by the `label` parameter; otherwise, it returns `0` (`FALSE`).

  For example, to apply a Data Redaction policy only in cases where the session label for the policy `hr_ols_pol` does not dominate nor is equal to label `hs`:

  ```
  expression  => 'OLS_LABEL_DOMINATES (''hr_ols_pol'',''hs'') = 0'
  ```

Related Topics

- *Oracle Label Security Administrator’s Guide*

3.5.7 Applying the Redaction Policy to All Users

You can apply the policy irrespective of the context to any user, with no filtering.

However, be aware that user `SYS` and users who have the `EXEMPT REDACTION POLICY` system or schema privilege are always exempt from Oracle Data Redaction policies.

- To apply the policy to users who are not `SYS` or have been granted the `EXEMPT REDACTION POLICY` privilege, write the `DBMS_REDACT.ADD_POLICY` `expression` parameter to evaluate to `TRUE`.

  For example:

  ```
  expression  => '1=1'
  ```

  With policy expression as above, Oracle Database performs the redaction but does not evaluate the policy expression. As a result, the performance of fetching from the target column is improved.

Related Topics

- Exemption of Users from Oracle Data Redaction Policies
  You can exempt users from having Oracle Data Redaction policies applied to the data they access.

3.6 Creating and Managing Multiple Named Policy Expressions

A named, centrally managed Oracle Data Redaction policy expression can be used in multiple redaction policies and applied to multiple tables or views.

3.6.1 About Data Redaction Policy Expressions to Define Conditions

A Oracle Data Redaction named policy expression is designed to work as an alternative to the policy expression that is used in existing Data Redaction policies.

A policy expression enables you to redact data based on runtime conditions. This type of policy expression can affect whether or not redaction takes place on columns of the table or view on which the redaction policy is defined. By default, a Data Redaction policy expression applies to all the columns that belong to the Data Redaction policy defined on that table or view. Alternatively, you can choose to create and associate a policy expression for individual columns of a table or view. These column level policy expressions are called named policy expressions. The column to which you apply a named policy expression must already be redacted by a Data Redaction policy.

You can use Data Redaction named policy expressions in the following ways:

- Each named policy expression can be associated with multiple columns of the same or different tables or views.
- Each named policy expression can be associated with columns within the same or different Data Redaction policies.
- The named policy expression overrides the default policy expression of the associated columns. The default policy expression still applies to columns that have no named policy expressions applied to them.
- Any updates made to a named policy expression apply to all of the column associations of the expression.

- You cannot associate multiple named policy expressions for the same column.
- You cannot associate named policy expressions with columns in a different pluggable database (PDB).

Table 3-8 describes the `DBMS_REDACT` PL/SQL procedures that you can use to create and manage named policy expressions. To find information about policy expressions, query the `REDACTION_EXPRESSIONS` data dictionary view.

Table 3-8 DBMS_REDACT Policy Expression Procedures

Procedure	Description
`DBMS_REDACT.CREATE_POLICY_EXPRESSION`	Creates a Data Redaction named policy expression
`DBMS_REDACT.UPDATE_POLICY_EXPRESSION`	Updates a Data Redaction named policy expression
`DBMS_REDACT.APPLY_POLICY_EXPR_TO_COL`	Applies a Data Redaction named policy expression to a table or a view column
`DBMS_REDACT.DROP_POLICY_EXPRESSION`	Drops a Data Redaction named policy expression

Related Topics

- Managing Data Redaction Named Policy Expressions Using Enterprise Manager
  You can manage Oracle Data Redaction named policy expressions in Enterprise Manager Cloud Control.

3.6.2 Creating and Applying a Data Redaction Named Policy Expression

The `DBMS_REDACT.CREATE_POLICY_EXPRESSION` and `DBMS_REDACT.APPLY_POLICY_EXPR_TO_COL` enable you to create and apply a Data Redaction named policy expression.

1. Connect to the PDB as a user who has the `EXECUTE` privilege on the `DBMS_REDACT` PL/SQL package and the `ADMINISTER REDACTION POLICY` system or schema privilege.
2. Ensure that the `COMPATIBLE` initialization parameter is set to `12.2.0.0` or later.

   To find the current `COMPATIBLE` setting, use the `SHOW PARAMETER` command.
3. To create the named policy expression, run the `DBMS_REDACT.CREATE_POLICY_EXPRESSION` procedure.

   For example:

```
BEGIN
 DBMS_REDACT.CREATE_POLICY_EXPRESSION (
  policy_expression_name        => 'redact_pol',
  expression                    => '1=1',
  policy_expression_description => 'Determines whether the column will be redacted');
END;
/
```

4. Run the `DBMS_REDACT.APPLY_POLICY_EXPR_TO_COL` procedure to apply the named policy expression to a table or view column.

   For example, assume that you have already created a Data Redaction policy on the `SALARY` column of the `HR.EMPLOYEES` table, as follows:

```
BEGIN
  DBMS_REDACT.ADD_POLICY  (
   object_schema           => 'hr',
   object_name             => 'employees',
   policy_name             => 'overall_policy',
   expression              => '1=0');
END;
/
BEGIN
  DBMS_REDACT.ALTER_POLICY (
   object_schema           => 'hr',
   object_name             => 'employees' ,
   policy_name             => 'overall_policy',
   function_type           => DBMS_REDACT.FULL,
   action                  => DBMS_REDACT.ADD_COLUMN,
   column_name             => 'salary');
END;
/
```

Then you can apply the policy expression to the `SALARY` column as follows:

```
BEGIN
 DBMS_REDACT.APPLY_POLICY_EXPR_TO_COL (
   object_schema           => 'hr',
   object_name             => 'employees',
   column_name             => 'salary',
   policy_expression_name  => 'redact_pol');
END;
/
```

In this specification:

- `object_schema`: Specifies the schema of the object on which named policy expression will be applied. If you omit this setting (or enter `NULL`), then Oracle Database uses the name of the current schema.
- `object_name`: Specifies the name of the table or view to be used for the named policy expression.
- `column_name`: Specifies the column to which you want to apply the named policy expression.
- `policy_expression_name`: Specifies the name of the named policy expression.

3.6.3 Updating a Data Redaction Named Policy Expression

You can use the `DBMS_REDACT.UPDATE_POLICY_EXPRESSION` procedure to update a Data Redaction named policy expression. The update takes place immediately and is reflected in all the columns that use the named policy expression.

You can query the `REDACTION_EXPRESSIONS` data dictionary view to find existing Data Redaction policy expressions.

1. Connect to the PDB as a user who has the `EXECUTE` privilege on the `DBMS_REDACT` PL/SQL package and the `ADMINISTER REDACTION POLICY` system or schema privilege.
2. Ensure that the `COMPATIBLE` initialization parameter is set to `12.2.0.0` or later.

   To find the current `COMPATIBLE` setting, use the `SHOW PARAMETER` command.
3. Run the `DBMS_REDACT.UPDATE_POLICY_EXPRESSION` procedure to perform the update.

   For example:

```
BEGIN
 DBMS_REDACT.UPDATE_POLICY_EXPRESSION(
  policy_expression_name => 'redact_pol',
  expression              => '1=0');
END;
/
```

3.6.4 Dropping a Data Redaction Named Policy Expression

You can use the `DBMS_REDACT.DROP_POLICY_EXPRESSION` procedure to drop a Data Redaction named policy expression.

You can query the `REDACTION_EXPRESSIONS` data dictionary view to find the existing Data Redaction named policy expressions.

1. Connect to the PDB as a user who has the `EXECUTE` privilege on the `DBMS_REDACT` PL/SQL package and the `ADMINISTER REDACTION POLICY` system or schema privilege.
2. Ensure that the `COMPATIBLE` initialization parameter is set to `12.2.0.0` or later.

   To find the current `COMPATIBLE` setting, use the `SHOW PARAMETER` command.
3. Remove the named policy expression's association with any table or view column.

   You cannot drop a named policy expression if it is associated with an existing table or view column. To remove a given column's association with a named policy expression (to revert to redacting that column based on the evaluation result of the default policy expression), you must set the `policy_expression_name` parameter of the `DBMS_REDACT.APPLY_POLICY_EXPR_TO_COL` procedure to `NULL`.

   For example:

   ```
   BEGIN
    DBMS_REDACT.APPLY_POLICY_EXPR_TO_COL(
      object_schema          => 'hr',
      object_name            => 'employees',
      column_name            => 'salary',
      policy_expression_name =>  null);
   END;
   /
   ```
4. Run `DBMS_REDACT.DROP_POLICY_EXPRESSION` to drop the policy expression.

   For example:

   ```
   BEGIN
    DBMS_REDACT.DROP_POLICY_EXPRESSION(
     policy_expression_name  => 'redact_pol');
   END;
   /
   ```

3.6.5 Tutorial: Creating and Sharing a Data Redaction Named Policy Expression

This tutorial shows how to create an Oracle Data Redaction named policy expression, apply it to multiple tables, and centrally manage the named policy expression.

3.6.5.1 Step 1: Create Users for This Tutorial

You must create two users for this tutorial: `dr_admin`, who will create the Oracle Data Redaction policies, and `hr_clerk`, who will test them.

Before you begin this tutorial, ensure that the `COMPATIBLE` initialization parameter is set to `12.2.0.0` or later. You can check this setting by using the `SHOW PARAMETER` command.

1. Connect to a PDB as user `SYS` with the `SYSDBA` administrative privilege.
2. Create the `dr_admin` and `hr_clerk` user accounts.

   ```
   GRANT CREATE SESSION TO dr_admin IDENTIFIED BY password;
   GRANT CREATE SESSION TO hr_clerk IDENTIFIED BY password;
   ```
3. Grant the following privileges to the `dr_admin` user:

   ```
   GRANT EXECUTE ON DBMS_REDACT TO dr_admin;
   GRANT ADMINISTER REDACTION POLICY TO dr_admin;
   ```
4. Connect as user `HR`.
5. Grant `hr_clerk` the `SELECT` privilege on the `EMPLOYEES` and `JOBS` tables.

   ```
   GRANT SELECT on EMPLOYEES to hr_clerk;
   GRANT SELECT on JOBS to hr_clerk;
   ```

3.6.5.2 Step 2: Create an Oracle Data Redaction Policy

User `dr_admin` is ready to create an Oracle Data Redaction policy to protect the `HR.EMPLOYEES` and `HR.JOBS` tables.

1. Connect to the PDB as user `dr_admin`.
2. Create the `hr_emp_redact_comp_pol` policy, which will perform full redaction of the `HR.EMPLOYEES.SALARY` column.

   ```
   BEGIN
    DBMS_REDACT.ADD_POLICY(
      object_schema       => 'hr',
      object_name         => 'employees',
      column_name         => 'salary',
      policy_name         => 'hr_emp_redact_comp_pol',
      function_type       => DBMS_REDACT.FULL,
      expression          => '1=1');
   END;
   /
   ```
3. Alter the `hr_redact_comp_pol` policy to also redact the `COMMISSION_PCT` column of the `HR.EMPLOYEES` table.

   ```
   BEGIN
    DBMS_REDACT.ALTER_POLICY(
      object_schema       => 'hr',
      object_name         => 'employees',
      policy_name         => 'hr_emp_redact_comp_pol',
      action              => DBMS_REDACT.ADD_COLUMN,
      column_name         => 'commission_pct',
      function_type       => DBMS_REDACT.FULL);
   END;
   /
   ```

4. Create the `hr_jobs_redact_comp_pol` policy for the `max_salary` column of the `HR.JOBS` table.

   ```
   BEGIN
    DBMS_REDACT.ADD_POLICY(
      object_schema       => 'hr',
      object_name         => 'jobs',
      column_name         => 'max_salary',
      policy_name         => 'hr_jobs_redact_comp_pol',
      function_type       => DBMS_REDACT.FULL,
      expression          => '1=1');
   END;
   /
   ```

At this stage, the data in the `HR.EMPLOYEES.SALARY`, `HR.EMPLOYEES.COMMISSION_PCT`, and `HR.JOBS.MAX_SALARY` columns are redacted.

3.6.5.3 Step 3: Test the Oracle Data Redaction Policy

User `hr_clerk` is ready to query the tables that have redacted data.

1. Connect as user `hr_clerk`.
2. Query the `HR.EMPLOYEES` table.

   ```
   SELECT SALARY, COMMISSION_PCT FROM HR.EMPLOYEES WHERE SALARY > 15000;
   ```

   The output should be as follows:

   ```
       SALARY COMMISSION_PCT
   ---------- --------------
            0
            0
            0
   ```

3. Query the `HR.JOBS` table.

   ```
   SELECT MAX_SALARY FROM HR.JOBS WHERE MAX_SALARY > 15000;
   ```

   The output should be as follows:

   ```
   MAX_SALARY
   ----------
            0
            0
            0
            0
            0
   ```

3.6.5.4 Step 4: Create and Apply a Named Policy Expression to the Redacted Table Columns

Next, user `dr_admin` is ready to create a Data Redaction named policy expression (`hr_redact_pol`) and apply it to two of the three redacted table columns.

This policy expression will enable user `hr_clerk` to view the redacted data.

1. Connect as user `dr_admin`.
2. Create the policy expression.

   ```
   BEGIN
    DBMS_REDACT.CREATE_POLICY_EXPRESSION(
   ```

```
   policy_expression_name  => 'hr_redact_pol',
   expression              => 'SYS_CONTEXT(''USERENV'',''SESSION_USER'') !=
''HR_CLERK''');
END;
/
```

This expression returns FALSE for the hr_clerk user, which enables the hr_clerk user to view actual data in the HR.EMPLOYEES and HR.JOBS tables that are subject to the Data Redaction policies.

3. Apply the hr_redact_pol policy expression to the HR.EMPLOYEES.SALARY column.

```
BEGIN
 DBMS_REDACT.APPLY_POLICY_EXPR_TO_COL(
   object_schema          => 'hr',
   object_name            => 'employees',
   column_name            => 'salary',
   policy_expression_name => 'hr_redact_pol');
END;
/
```

4. Apply the hr_redact_pol policy expression to the HR.JOBS.MAX_SALARY column.

```
BEGIN
 DBMS_REDACT.APPLY_POLICY_EXPR_TO_COL(
   object_schema          => 'hr',
   object_name            => 'jobs',
   column_name            => 'max_salary',
   policy_expression_name => 'hr_redact_pol');
END;
/
```

User hr_clerk can view data in the HR.EMPLOYEES.SALARY and HR.JOBS.MAX_SALARY, but the data in the HR.EMPLOYEES.COMMISSION_PCT column will still be redacted for this user.

3.6.5.5 Step 5: Test the Data Redaction Named Policy Expression

User hr_clerk is now ready to test the hr_redact_pol named policy expression.

1. Connect as user hr_clerk.
2. Query the HR.EMPLOYEES table.

   ```
   SELECT SALARY, COMMISSION_PCT FROM HR.EMPLOYEES WHERE SALARY > 15000;
   ```

   The output should be as follows:

   ```
       SALARY COMMISSION_PCT
   ---------- --------------
        24000
        17000
        17000
   ```

   User hr_clerk now can view the SALARY column data, but still has no access to the COMMISSION_PCT column data.
3. Query the HR.JOBS table.

   ```
   SELECT MAX_SALARY FROM HR.JOBS WHERE MAX_SALARY > 15000;
   ```

   The output should be as follows:

   ```
   MAX_SALARY
   ----------
   ```

```
     40000
     30000
     16000
     16000
     20080
```

User `hr_clerk` now can view the `MAX_SALARY` column data.

3.6.5.6 Step 6: Modify the Data Redaction Named Policy Expression

User `dr_admin` decides to modify the Data Redaction named policy expression so that user `hr_clerk` will get redacted data, not user `HR`.

1. Connect as user `dr_admin`.
2. Modify the `hr_redact_pol` policy as follows:

```
BEGIN
 DBMS_REDACT.UPDATE_POLICY_EXPRESSION(
  policy_expression_name => 'hr_redact_pol',
  expression             => 'SYS_CONTEXT(''USERENV'',''SESSION_USER'') != ''HR''');
END;
/
```

3.6.5.7 Step 7: Test the Modified Named Policy Expression

Users `HR` and `hr_clerk` are ready to test the modified Data Redaction named policy expression.

1. Connect as user `HR`.
2. Query the `HR.EMPLOYEES` table.

```
SELECT SALARY, COMMISSION_PCT FROM HR.EMPLOYEES WHERE SALARY > 15000;
```

The output should be as follows:

```
    SALARY COMMISSION_PCT
---------- --------------
     24000
     17000
     17000
```

User `HR` now has access to the actual data. A query by `HR` on the `HR.JOBS.MAX_SALARY` column will produce similar results.

```
SELECT MAX_SALARY FROM HR.JOBS WHERE MAX_SALARY > 15000;

MAX_SALARY
----------
     40000
     30000
     16000
     16000
     20080
```

3. Connect as user `hr_clerk`.
4. Query the `HR.EMPLOYEES` and `HR.JOBS` tables and then observe the results.

```
SELECT SALARY, COMMISSION_PCT FROM HR.EMPLOYEES WHERE SALARY > 15000;

    SALARY COMMISSION_PCT
```

```
---------- --------------
         0
         0
         0

SELECT MAX_SALARY FROM HR.JOBS WHERE MAX_SALARY > 15000;

MAX_SALARY
----------
         0
         0
         0
         0
         0
```

3.6.5.8 Step 8: Remove the Components of This Tutorial

If you do not need the components of this tutorial, then you can remove them.

1. Connect as user `dr_admin`.
2. Modify the policy expression so that it is no longer associated with the table columns that are associated with the expression.

   To do so, you must set the `policy_expression_name` parameter to `NULL`.

   ```
   BEGIN
    DBMS_REDACT.APPLY_POLICY_EXPR_TO_COL(
      object_schema          => 'hr',
      object_name            => 'employees',
      column_name            => 'salary',
      policy_expression_name =>  null);
   END;
   /

   BEGIN
    DBMS_REDACT.APPLY_POLICY_EXPR_TO_COL(
      object_schema          => 'hr',
      object_name            => 'jobs',
      column_name            => 'max_salary',
      policy_expression_name =>  null);
   END;
   /
   ```
3. Drop the policy expression.

   ```
   BEGIN
    DBMS_REDACT.DROP_POLICY_EXPRESSION(
     policy_expression_name  => 'hr_redact_pol');
   END;
   /
   ```
4. Drop the `hr_emp_redact_comp_pol` and `hr_jobs_redact_comp_pol` Data Redaction policies.

   ```
   BEGIN
     DBMS_REDACT.DROP_POLICY (
       object_schema  => 'hr',
       object_name    => 'employees',
       policy_name    => 'hr_emp_redact_comp_pol');
   END;
   /

   BEGIN
   ```

```
  DBMS_REDACT.DROP_POLICY (
    object_schema  => 'hr',
    object_name    => 'jobs',
    policy_name    => 'hr_jobs_redact_comp_pol');
END;
/
```

5. Connect as the `SYSTEM` user or a user who has privileges to drop user accounts.
6. Drop the `dr_admin` and `hr_clerk` user accounts.

```
DROP USER dr_admin;
DROP USER hr_clerk;
```

3.7 Creating a Full Redaction Policy and Altering the Full Redaction Value

You can create a full redaction policy to redact all contents in a data column, and optionally, you can alter the default full redaction value.

3.7.1 Creating a Full Redaction Policy

A full data redaction policy redacts all the contents of a data column.

3.7.1.1 About Creating Full Data Redaction Policies

To set a redaction policy to redact all data in the column, you must set the `function_type` parameter to `DBMS_REDACT.FULL`.

By default, `NUMBER` data type columns are replaced with zero (`0`) and character data type columns are replaced with a single space ( ). You can modify this default by using the `DBMS_REDACT.UPDATE_FULL_REDACTION_VALUES` procedure.

Related Topics

- Altering the Default Full Data Redaction Value
  The `DBMS_REDACT.UPDATE_FULL_REDACTION_VALUES` procedure alters the default full data redaction value.

3.7.1.2 Syntax for Creating a Full Redaction Policy

The `DBMS_REDACT.ADD_POLICY` procedure enables you to create a full redaction policy.

The `DBMS_REDACT.ADD_POLICY` fields for creating a full data redaction policy are as follows:

```
DBMS_REDACT.ADD_POLICY (
   object_schema            IN VARCHAR2 := NULL,
   object_name              IN VARCHAR2,
   column_name              IN VARCHAR2 := NULL,
   policy_name              IN VARCHAR2,
   function_type            IN BINARY_INTEGER := NULL,
   expression               IN VARCHAR2,
   enable                   IN BOOLEAN := TRUE);
```

In this specification:

- `object_schema`, `object_name`, `column_name`, `policy_name`, `expression`, `enable`: See General Syntax of the DBMS_REDACT.ADD_POLICY Procedure.

- `function_type`: Specifies the function used to set the type of redaction. Enter `DBMS_REDACT.FULL`.

  If you omit the `function_type` parameter, then the default redaction `function_type` setting is `DBMS_REDACT.FULL`.

  Remember that the data type of the column determines which `function_type` settings that you are permitted to use. See Comparison of Full, Partial, Regexp, Random, and Nullify Redaction Based on Data Types.

3.7.1.3 Example: Full Redaction Policy

You can use the `DBMS_REDACT.ADD_POLICY` PL/SQL procedure to create a full redaction policy.

Example 3-1 shows how to use full redaction for all the values in the `HR.EMPLOYEES` table `COMMISSION_PCT` column. Because the expression parameter in this example always evaluates to `TRUE`, the data redaction policy is applied to any user querying the table except for users who have been granted the `EXEMPT REDACTION POLICY` system or schema privilege, or the `SYSDBA` administrative privilege.

Example 3-1 Full Data Redaction Policy

```
BEGIN
 DBMS_REDACT.ADD_POLICY(
   object_schema    => 'hr',
   object_name      => 'employees',
   column_name      => 'commission_pct',
   policy_name      => 'redact_com_pct',
   function_type    => DBMS_REDACT.FULL,
   expression       => '1=1');
END;
/
```

Query and redacted result:

```
SELECT COMMISSION_PCT FROM HR.EMPLOYEES;

COMMISSION_PCT
--------------
0
0
0
```

Related Topics

- Exemption of Users from Oracle Data Redaction Policies
  You can exempt users from having Oracle Data Redaction policies applied to the data they access.

3.7.1.4 Example: Fully Redacted Character Values

You can use the `DBMS_REDACT.ADD_POLICY` PL/SQL procedure to create a policy that fully redacts character values.

Example 3-2 shows how to fully redact the user IDs of the `user_id` column in the `mavis.cust_info` table. The `user_id` column is of the `VARCHAR2` data type. The output is a single space. The `expression` setting enables users who have the `MGR` role to view the user IDs.

Example 3-2   Fully Redacted Character Values

```
BEGIN
 DBMS_REDACT.ADD_POLICY(
   object_schema    => 'mavis',
   object_name      => 'cust_info',
   column_name      => 'user_id',
   policy_name      => 'redact_cust_user_ids',
   function_type    => DBMS_REDACT.FULL,
   expression       => 'SYS_CONTEXT(''SYS_SESSION_ROLES'',''MGR'') = ''FALSE''');
END;
/
```

Query and redacted result:

```
SELECT user_id FROM mavis.cust_info;

USER_ID
------------
0
0
0
```

3.7.2 Altering the Default Full Data Redaction Value

The `DBMS_REDACT.UPDATE_FULL_REDACTION_VALUES` procedure alters the default full data redaction value.

3.7.2.1 About Altering the Default Full Data Redaction Value

You can alter the default displayed values for full Data Redaction polices.

By default, `0` is the redacted value when Oracle Database performs full redaction (`DBMS_REDACT.FULL`) on a column of the `NUMBER` data type. If you want to change it to another value (for example, `7`), then you can run the `DBMS_REDACT.UPDATE_FULL_REDACTION_VALUES` procedure to modify this value. The modification applies to all of the Data Redaction policies in the current database instance. After you modify a value, you must restart the database for it to take effect. You can find the current values by querying the `REDACTION_VALUES_FOR_TYPE_FULL` data dictionary view.

Be aware that this change affects all Data Redaction policies in the database that use full data redaction. Before you alter the default full data redaction value, examine the affect that this change would have on existing full Data Redaction policies.

3.7.2.2 Syntax for the DBMS_REDACT.UPDATE_FULL_REDACTION_VALUES Procedure

The `DBMS_REDACT.UPDATE_FULL_REDACTION_VALUES` procedure accommodates the standard supported Oracle Database data types.

The syntax is as follows:

```
DBMS_REDACT.UPDATE_FULL_REDACTION_VALUES (
 number_val       IN NUMBER                    NULL,
 binfloat_val     IN BINARY_FLOAT              NULL,
 bindouble_val    IN BINARY_DOUBLE             NULL,
 char_val         IN CHAR                      NULL,
 varchar_val      IN VARCHAR2                  NULL,
 nchar_val        IN NCHAR                     NULL,
 nvarchar_val      IN NVARCHAR2                  NULL,
 date_val          IN DATE                       NULL,
 ts_val            IN TIMESTAMP                  NULL,
 tswtz_val         IN TIMESTAMP WITH TIME ZONE  NULL,
 blob_val          IN BLOB                       NULL,
 clob_val          IN CLOB                       NULL,
 nclob_val         IN NCLOB                      NULL,
 boolean_val       IN BOOLEAN                    NULL);
```

In this specification:

- `number_val` modifies the default value for columns of the `NUMBER` data type.
- `binfloat_val` modifies the default value for columns of the `BINARY_FLOAT` data type.
- `bindouble_val` modifies the default value for columns of the `BINARY_DOUBLE` data type.
- `char_val` modifies the default value for columns of the `CHAR` data type.
- `varchar_val` modifies the default value for columns of the `VARCHAR2` data type.
- `nchar_val` modifies the default value for columns of the `NCHAR` data type.
- `nvarchar_val` modifies the default value for columns of the `NVARCHAR2` data type.
- `date_val` modifies the default value for columns of the `DATE` data type.
- `ts_val` modifies the default value for columns of the `TIMESTAMP` data type.
- `tswtz_val` modifies the default value for columns of the `TIMESTAMP WITH TIME ZONE` data type.
- `blob_val` modifies the default value for columns of the `BLOB` data type.
- `clob_val` modifies the default value for columns of the `CLOB` data type.
- `nclob` modifies the default value for columns of the `NCLOB` data type.
- `boolean_val` modifies the default value for columns of the `BOOLEAN` data type.

  The first time `DBMS_REDACT.UPDATE_FULL_REDACTION_VALUES` is invoked with a non-`NULL` value for its `boolean_val` parameter after the `COMPATIBLE` database initialization parameter is set to 23 or higher, you will see the `BOOLEAN_VALUE` column appear in the catalog view for `REDACTION_VALUES_FOR_TYPE_FULL` and the `BOOLCOL` column appear in the `SYS.RADM_FPTM$` data dictionary table.

3.7.2.3 Modifying the Default Full Data Redaction Value

To modify the default full data redaction value, use the `DBMS_REDACT.UPDATE_FULL_REDACTION_VALUES` procedure.

1. Connect to the PDB as a user who has the `EXECUTE` privilege on the `DBMS_REDACT` PL/SQL package and the `ADMINISTER REDACTION POLICY` system or schema privilege.
2. Check the value that you want to change.

   For example, to check the current value for columns that use the `NUMBER` data type:

   ```
   SELECT NUMBER_VALUE FROM REDACTION_VALUES_FOR_TYPE_FULL;

   NUMBER_VALUE
   ------------
              0
   ```

3. Run the `DBMS_REDACT.UPDATE_FULL_REDACTION_VALUES` procedure to modify the value.

For example:

```
EXEC DBMS_REDACT.UPDATE_FULL_REDACTION_VALUES (number_val => 7);
```

4. Close and then re-open the PDB.

```
ALTER PLUGGABLE DATABASE pdb_name CLOSE IMMEDIATE;
ALTER PLUGGABLE DATABASE pdb_name OPEN;
```

5. Check the value that you changed.

```
SELECT NUMBER_VALUE FROM REDACTION_VALUES_FOR_TYPE_FULL;

NUMBER_VALUE
------------
           7
```

3.8 Creating a Partial Redaction Policy

In partial data redaction, you can redact portions of data for different kinds of data types.

3.8.1 About Creating Partial Redaction Policies

In partial data redaction, only a portion of the data, such as the first five digits of an identification number, are redacted.

For example, you can redact most of a credit card number with asterisks (*), except for the last 4 digits. You can create policies for columns that use character, number, or date-time data types. For policies that redact character data types, you can use fixed character redaction formats. If you have the Enterprise Manager for Oracle Database 12.1.0.7 plug-in deployed on your system, then you can also create and save custom redaction formats.

Note:

In previous releases, the term shortcut was used for the term format.

3.8.2 Syntax for Creating a Partial Redaction Policy

The `DBMS_REDACT.ADD_POLICY` statement enables you to create policies that redact specific parts of the data returned to the application.

The `DBMS_REDACT.ADD_POLICY` fields for creating a partial redaction policy are as follows:

```
DBMS_REDACT.ADD_POLICY (
   object_schema            IN VARCHAR2 := NULL,
   object_name              IN VARCHAR2,
   column_name              IN VARCHAR2 := NULL,
   policy_name              IN VARCHAR2,
   function_type            IN BINARY_INTEGER := NULL,
   function_parameters      IN VARCHAR2 := NULL,
   expression               IN VARCHAR2,
   enable                   IN BOOLEAN := TRUE);
```

In this specification:

- `object_schema`, `object_name`, `column_name`, `policy_name`, `expression`, `enable`: See General Syntax of the DBMS_REDACT.ADD_POLICY Procedure

- `function_type`: Specifies the function used to set the type of redaction. Enter `DBMS_REDACT.PARTIAL`.
- `function_parameters`: The parameters that you set here depend on the data type of the column specified for the `column_name` parameter. See the following sections for details:
  - Creating Partial Redaction Policies Using Fixed Character Formats
  - Creating Partial Redaction Policies Using Character Data Types
  - Creating Partial Redaction Policies Using Number Data Types
  - Creating Partial Redaction Policies Using Date-Time Data Types

3.8.3 Creating Partial Redaction Policies Using Fixed Character Formats

The `DBMS_REDACT.ADD_POLICY` `function_parameters` parameter can be configured to support redaction of fixed character data types.

3.8.3.1 Settings for Fixed Character Formats

Oracle Data Redaction provides special predefined formats to configure policies that use fixed characters.

Table 3-9 describes `DBMS_REDACT.ADD_POLICY` `function_parameters` parameter formats that you can use for commonly redacted identity numbers (such as Social Security Numbers or Canadian Social Insurance Numbers), postal codes, and credit card numbers that use either the `VARCHAR2` or `NUMBER` data types for their columns.

Table 3-9 Partial Fixed Character Redaction Formats

Format	Description
`DBMS_REDACT.REDACT_US_SSN_F5`	Redacts the first 5 numbers of Social Security Numbers when the column is a `VARCHAR2` data type. For example, the number `987-65-4320` becomes `XXX-XX-4320`.
`DBMS_REDACT.REDACT_US_SSN_L4`	Redacts the last 4 numbers of Social Security Numbers when the column is a `VARCHAR2` data type. For example, the number `987-65-4320` becomes `987-65-XXXX`.
`DBMS_REDACT.REDACT_US_SSN_ENTIRE`	Redacts the entire Social Security Number when the column is a `VARCHAR2` data type. For example, the number `987-65-4320` becomes `XXX-XX-XXXX`.
`DBMS_REDACT.REDACT_NUM_US_SSN_F5`	Redacts the first 5 numbers of Social Security Numbers when the column is a `NUMBER` data type. For example, the number `987654320` becomes `XXXXX4320`.
`DBMS_REDACT.REDACT_NUM_US_SSN_L4`	Redacts the last 4 numbers of Social Security Numbers when the column is a `NUMBER` data type. For example, the number `987654320` becomes `98765XXXX`.
`DBMS_REDACT.REDACT_NUM_US_SSN_ENTIRE`	Redacts the entire Social Security Number when the column is a `NUMBER` data type. For example, the number `987654320` becomes `XXXXXXXXX`.

Table 3-9 (Cont.) Partial Fixed Character Redaction Formats

Format	Description
`DBMS_REDACT.REDACT_SIN_NUMBER`	Redacts the Canadian Social Insurance number by replacing the first 6 digits by `9` (number). For example, `123456789` is redacted to `999999789`.
`DBMS_REDACT.REDACT_SIN_UNFORMATTED`	Redacts the Canadian Social Insurance number by replacing the first 6 digits by `X` (string). For example, `123456789` is redacted to `XXXXXX789`.
`DBMS_REDACT.REDACT_SIN_FORMATTED`	Redacts the Canadian Social Insurance Number by replacing the first 6 digits by `X` (string). For example, `123-456-789` is redacted to `XXX-XXX-789`.
`DBMS_REDACT.REDACT_UK_NIN_FORMATTED`	Redacts the UK National Insurance number by replacing the first 6 digits by `X` (string) but leaving the alphabetic characters as is. For example, `ET 27 02 23 D` is redacted to `ET XX XX XX D`.
`DBMS_REDACT.REDACT_UK_NIN_UNFORMATTED`	Redacts the UK National Insurance number by replacing the first 6 digits by `X` (string) and leaving the alphabetic characters as is. For example, `ET270223D` is redacted to `ETXXXXXXD`.
`DBMS_REDACT.REDACT_CCN_FORMATTED`	Redacts the credit card number (other than American Express) by replacing everything but the last 4 digits by `*`. For example, the credit card number `5105-1051-0510-5100` is redacted to `****–****–****–5100`.
`DBMS_REDACT.REDACT_CCN_NUMBER`	Redacts the credit card number (other than American Express) by replacing everything but the last 4 digits by `9`. For example, the credit card number `5105105105105100` is redacted to `9999999999995100`. Note that the last four digits are not redacted.
`DBMS_REDACT.REDACT_CCN16_F12`	Redacts a 16-digit credit card number (other than American Express), leaving the last 4 digits displayed. For example, `5105 1051 0510 5100` becomes `****-****-****-5100`.
`DBMS_REDACT.REDACT_AMEX_CCN_FORMATTED`	Redacts the American Express credit card number by replacing the digits with `*` except the last 5 digits. For example, the credit card number `3782 822463 10005` is redacted to `**** ****** 10005`.
`DBMS_REDACT.REDACT_AMEX_CCN_NUMBER`	Redacts the American Express Credit Card Number by replacing the digits with `0` except the last 5 digits. For example, the credit card number `3782 822463 10005` is redacted to `0000 000000 10005`.
`DBMS_REDACT.REDACT_ZIP_CODE`	Redacts a 5-digit postal code when the column is a `VARCHAR2` data type. For example, `95476` becomes `XXXXX`.
`DBMS_REDACT.REDACT_NUM_ZIP_CODE`	Redacts a 5-digit postal code when the column is a `NUMBER` data type. For example, `95476` becomes `99999`.
`DBMS_REDACT.REDACT_DATE_EPOCH`	Redacts all dates to `01-JAN-70`.

Table 3-9 (Cont.) Partial Fixed Character Redaction Formats

Format	Description
`DBMS_REDACT.REDACT_NA_PHONE_FORMATTED`	Redacts the North American phone number by leaving the area code, but replacing everything else with `X`. For example, `650-555-0100` is redacted to `650-XXX-XXXX`.
`DBMS_REDACT.REDACT_NA_PHONE_NUMBER`	Redacts the North American phone number by leaving the area code, but replacing everything else with `0`. For example, `6505550100` gets redacted to `6500000000`.
`DBMS_REDACT.REDACT_NA_PHONE_UNFORMATTED`	Redacts the North American phone number by leaving the area code, but replacing everything else with `X`. For example, `6505550100` is redacted to `650XXXXXXX`.
`DBMS_REDACT.REDACT_DATE_MILLENNIUM`	Redacts dates that are in the `DD-MON-YY` format to `01-JAN-00` (January 1, 2000).

Related Topics

- General Syntax of the DBMS_REDACT.ADD_POLICY Procedure
  To create a Data Redaction policy, you must use the `DBMS_REDACT.ADD_POLICY` procedure.

3.8.3.2 Example: Partial Redaction Policy Using a Fixed Character Format

You can use the `DBMS_REDACT.ADD_POLICY` PL/SQL procedure to create a partial redaction policy that uses a fixed character format.

Example 3-3 shows how Social Security Numbers in a `VARCHAR2` data type column can be redacted using the `REDACT_US_SSN_F5` format.

Example 3-3 Partially Redacted Character Values

```
BEGIN
 DBMS_REDACT.ADD_POLICY(
   object_schema       => 'mavis',
   object_name         => 'cust_info',
   column_name         => 'ssn',
   policy_name         => 'redact_cust_ssns3',
   function_type       => DBMS_REDACT.PARTIAL,
   function_parameters => DBMS_REDACT.REDACT_US_SSN_F5,
   expression          => '1=1',
   policy_description  => 'Partially redacts 1st 5 digits in Social Security Numbers',
   column_description  => 'ssn contains Social Security Numbers');
END;
/
```

Query and redacted result:

```
SELECT ssn FROM mavis.cust_info;

SSN
-------
XXX-XX-4320
XXX-XX-4323
XXX-XX-4325
XXX-XX-4329
```

3.8.4 Creating Partial Redaction Policies Using Character Data Types

The `DBMS_REDACT.ADD_POLICY` `function_parameters` parameter can be configured to support redaction of character data types.

3.8.4.1 Settings for Character Data Types

Oracle Data Redaction provides special settings to configure policies that use character data types.

When you set the `DBMS_REDACT.ADD_POLICY` `function_parameters` parameter to define partial redaction of character data types, enter values for the following settings in the order shown. Separate each value with a comma.

Note:

Be aware that you must use a fixed width character set for the partial redaction. In other words, each character redacted must be replaced by another of equal byte length. If you want to use a variable-length character set (for example, UTF-8), then you must use a regular expression-based redaction.

The settings are as follows:

1. **Input format:** Defines how the data is currently formatted. Enter `V` for each character that potentially can be redacted, such as all of the digits in a credit card number. Enter `F` for each character that you want to format using a formatting character, such as hyphens or blank spaces in the credit card number. Ensure that each character has a corresponding `V` or `F` value. (The input format values are not case-sensitive.)
2. **Output format:** Defines how the displayed data should be formatted. Enter `V` for each character to be potentially redacted. Replace each `F` character in the input format with the character that you want to use for the displayed output, such as a hyphen. (The output format values are not case-sensitive.)
3. **Mask character:** Specifies the character to be used for the redaction. Enter a single character to use for the redaction, such as an asterisk (*).
4. **Starting digit position:** Specifies the starting `V` digit position for the redaction.
5. **Ending digit position:** Specifies the ending `V` digit position for the redaction. Do not include the `F` positions when you decide on the ending position value.

For example, the following setting redacts the first 12 `V` digits of the credit card number `5105 1051 0510 5100`, and replaces the `F` positions (which are blank spaces) with hyphens to format it in a style normally used for credit card numbers, resulting in `****-****-****-4320`.

```
function_parameters  => 'VVVVFVVVVFVVVVFVVVV,VVVV-VVVV-VVVV-VVVV,*,1,12',
```

Related Topics

- Syntax for Creating a Partial Redaction Policy
  The `DBMS_REDACT.ADD_POLICY` statement enables you to create policies that redact specific parts of the data returned to the application.

- General Syntax of the DBMS_REDACT.ADD_POLICY Procedure
  To create a Data Redaction policy, you must use the DBMS_REDACT.ADD_POLICY procedure.

3.8.4.2 Example: Partial Redaction Policy Using a Character Data Type

The DBMS_REDACT.ADD_POLICY PL/SQL procedure can create a partial redaction policy that uses a character data type.

Example 3-4 shows how to redact Social Security Numbers that are in a VARCHAR2 data type column and to preserve the character hyphens in the Social Security Number.

Example 3-4 Partially Redacted Character Values

```
BEGIN
 DBMS_REDACT.ADD_POLICY(
   object_schema       => 'mavis',
   object_name         => 'cust_info',
   column_name         => 'ssn',
   policy_name         => 'redact_cust_ssns2',
   function_type       => DBMS_REDACT.PARTIAL,
   function_parameters => 'VVVFVVFVVVV,VVV-VV-VVVV,*,1,5',
   expression          => '1=1',
   policy_description  => 'Partially redacts Social Security Numbers',
   column_description  => 'ssn contains character Social Security Numbers');
END;
/
```

Query and redacted result:

```
SELECT ssn FROM mavis.cust_info;

SSN
-----------
***-**-4320
***-**-4323
***-**-4325
***-**-4329
```

3.8.5 Creating Partial Redaction Policies Using Number Data Types

The DBMS_REDACT.ADD_POLICY function_parameters parameter can be configured to support redaction of number data types.

3.8.5.1 Settings for Number Data Types

When you set values for the number data type, you must specify a redact character, a starting digit position, and an ending digit position.

For partial redaction of number data types, you can enter values for the following settings for the function_parameters parameter of the DBMS_REDACT.ADD_POLICY procedure, in the order shown.

1. **Redact character:** Specifies the character to display. Enter a number from 0 to 9.
2. **Starting digit position:** Specifies the starting digit position for the redaction, such as 1 for the first digit.
3. **Ending digit position:** Specifies the ending digit position for the redaction.

For example, the following setting redacts the first five digits of the Social Security Number `987654321`, resulting in `999994321`.

```
function_parameters  => '9,1,5',
```

Related Topics

- General Syntax of the DBMS_REDACT.ADD_POLICY Procedure
  To create a Data Redaction policy, you must use the `DBMS_REDACT.ADD_POLICY` procedure.

3.8.5.2 Example: Partial Redaction Policy Using a Number Data Type

The `DBMS_REDACT.ADD_POLICY` procedure can create a partial redaction policy that uses a number data type.

Example 3-5 shows how to partially redact a set of Social Security Numbers in the `mavis.cust_info` table, for any application user who logs in. (Hence, the `expression` parameter evaluates to `TRUE`.)

This type of redaction is useful when the application is expecting a formatted number and not a string. In this scenario, the Social Security Numbers are in a column of the data type `NUMBER`. In other words, the `ssn` column contains numbers only, not other characters such as hyphens or blank spaces.

Example 3-5 Partially Redacted Data Redaction Numeric Values

```
BEGIN
 DBMS_REDACT.ADD_POLICY(
   object_schema       => 'mavis',
   object_name         => 'cust_info',
   column_name         => 'ssn',
   policy_name         => 'redact_cust_ssns1',
   function_type       => DBMS_REDACT.PARTIAL,
   function_parameters => '7,1,5',
   expression          => '1=1',
   policy_description  => 'Partially redacts Social Security Numbers',
   column_description  => 'ssn contains numeric Social Security Numbers');
END;
/
```

Query and redacted result:

```
SELECT ssn FROM mavis.cust_info;

SSN
---------
777774320
777774323
777774325
777774329
```

3.8.6 Creating Partial Redaction Policies Using Date-Time Data Types

The `DBMS_REDACT.ADD_POLICY function_parameters` parameter can be configured to support redaction of date-time data types.

3.8.6.1 Settings for Date-Time Data Types

Oracle Data Redaction provides special settings for configuring date-time data types.

For partial redaction of date-time data types, enter values for the following `DBMS_REDACT.ADD_POLICY` `function_parameters` parameter settings.

Enter these values in the order shown:

1. `m`: Redacts the month. To redact with a month name, append `1`–`12` to lowercase `m`. For example, `m5` displays as `MAY`. To omit redaction, enter an uppercase `M`.
2. `d`: Redacts the day of the month. To redact with a day of the month, append `1`–`31` to a lowercase `d`. For example, `d7` displays as `07`. If you enter a higher number than the days of the month (for example, `31` for the month of February), then the last day of the month is displayed (for example, `28`). To omit redaction, enter an uppercase `D`.
3. `y`: Redacts the year. To redact with a year, append `1`–`9999` to a lowercase `y`. For example, `y1984` displays as `84`. To omit redaction, enter an uppercase `Y`.
4. `h`: Redacts the hour. To redact with an hour, append `0`–`23` to a lowercase `h`. For example, `h20` displays as `20`. To omit redaction, enter an uppercase `H`.
5. `m`: Redacts the minute. To redact with a minute, append `0`–`59` to a lowercase `m`. For example, `m30` displays as `30`. To omit redaction, enter an uppercase `M`.
6. `s`: Redacts the second. To redact with a second, append `0`–`59` to a lowercase `s`. For example, `s45` displays as `45`. To omit redaction, enter an uppercase `S`.

Related Topics

- General Syntax of the DBMS_REDACT.ADD_POLICY Procedure
  To create a Data Redaction policy, you must use the `DBMS_REDACT.ADD_POLICY` procedure.

3.8.6.2 Example: Partial Redaction Policy Using Date-Time Data Type

The `DBMS_REDACT.ADD_POLICY` procedure can create a partial redaction policy that uses the date-time data type.

Example 3-6 shows how to partially redact a date. This example redacts the birth year of customers; replacing it with `13`, but retaining the remaining values.

Example 3-6 Partially Redacted Data Redaction Using Date-Time Values

```
BEGIN
 DBMS_REDACT.ADD_POLICY(
   object_schema       => 'mavis',
   object_name         => 'cust_info',
   column_name         => 'birth_date',
   policy_name         => 'redact_cust_bdate',
   function_type       => DBMS_REDACT.PARTIAL,
   function_parameters => 'MDy2013HMS',
   expression          => '1=1',
   policy_description  => 'Replaces birth year with 2013',
   column_description  => 'birth_date contains customer's birthdate');
END;
/
```

Query and redacted result:

```
SELECT birth_date FROM mavis.cust_info;

BIRTH_DATE
07-DEC-13 09.45.40.000000 AM
12-OCT-13 04.23.29.000000 AM
```

3.9 Creating a Regular Expression-Based Redaction Policy

A regular expression-based redaction policy enables you to redact data based on a search-and-replace model.

3.9.1 About Creating Regular Expression-Based Redaction Policies

Regular expression-based redaction enables you to search for patterns of data to redact.

For example, you can use regular expressions to redact email addresses, which can have varying character lengths. It is designed for use with character data only. You can use formats for the search and replace operation, or you can create custom pattern formats.

You cannot use regular expressions to redact a subset of the values in a column. The `REGEXP_PATTERN` (regular expression pattern) must match *all* of the values in order for the `REGEXP_REPLACE_STRING` setting to take effect, and the `REGEXP_REPLACE_STRING` must change the value.

For rows where the `REGEXP_PATTERN` fails to match, Data Redaction performs `DBMS_REDACT.FULL` redaction. This mitigates the risk of showing the actual data for those rows which it failed to match because of a mistake in the `REGEXP_PATTERN` causing the regular expression to fail to match all of the values in the column.

In addition, if no change to the value occurs as a result of the `REGEXP_REPLACE_STRING` setting during regular expression replacement operation, Data Redaction performs `DBMS_REDACT.FULL` redaction.

3.9.2 Syntax for Creating a Regular Expression-Based Redaction Policy

The `regexp_*` parameters of the `DBMS_REDACT.ADD_POLICY` procedure can create a regular expression-based redaction policy.

The `DBMS_REDACT.ADD_POLICY` fields for creating a regular expression-based data redaction policy are as follows:

```
DBMS_REDACT.ADD_POLICY (
   object_schema           IN VARCHAR2 := NULL,
   object_name             IN VARCHAR2,
   column_name             IN VARCHAR2 := NULL,
   policy_name             IN VARCHAR2,
   function_type           IN BINARY_INTEGER := NULL,
   expression              IN VARCHAR2,
   enable                  IN BOOLEAN := TRUE,
   regexp_pattern          IN VARCHAR2 := NULL,
   regexp_replace_string   IN VARCHAR2 := NULL,
   regexp_position         IN BINARY_INTEGER := 1,
   regexp_occurrence       IN BINARY_INTEGER := 0,
   regexp_match_parameter  IN VARCHAR2 := NULL);
```

In this specification:

- `object_schema`, `object_name`, `column_name`, `policy_name`, `expression`, `enable`: See General Syntax of the DBMS_REDACT.ADD_POLICY Procedure.
- `function_type`: Specifies the type of redaction. For regular expression based redaction, use either `DBMS_REDACT.REGEXP` or `DBMS_REDACT.REGEXP_WIDTH`.

  If you use the `DBMS_REDACT.REGEXP` redaction type, then no truncation occurs. This applies even if the redacted value is wider than the column width, and the Oracle Call Interface width attribute (`OCI_ATTR_CHAR_SIZE`) of the column is *not* preserved. (It becomes 4000, just as it does when the `REGEXP_REPLACE` SQL operator is used on a column.)

  Using the `DBMS_REDACT.REGEXP_WIDTH` redaction type truncates any redacted value that exceeds the width of the column, and ensures that the OCI width attribute of the column (`OCI_ATTR_CHAR_SIZE`) remains unchanged.

  Note the following:

  – Use the `DBMS_REDACT.REGEXP_WIDTH` function type if your applications depend on the value of the `OCI_ATTR_CHAR_SIZE` attribute. For example, applications that are built using the Oracle OLE DB Provider interface are sensitive to the value of the `OCI_ATTR_CHAR_SIZE` attribute. If you use `DBMS_REDACT.REGEXP` as the redaction type, then the `OCI_ATTR_CHAR_SIZE` always becomes 4000. This setting makes it unsuitable as the redaction type of policies on tables that are used by Oracle OLE DB based applications. See *Oracle Call Interface Developer's Guide* for more information about Oracle Call Interface parameter attributes.

  – When you set the `function_type` parameter to `DBMS_REDACT.REGEXP` or `DBMS_REDACT.REGEXP_WIDTH`, omit the `function_parameters` parameter from the `DBMS_REDACT.ADD_POLICY` procedure.

  – Specify the regular expression parameters in much the same way that you specify the `pattern`, `replace`, `position`, `occurrence`, and `match_parameter` arguments to the `REGEXP_REPLACE` SQL function. See *Oracle Database SQL Language Reference* for information about the `REGEXP_REPLACE` SQL function.

- `regexp_pattern`: Describes the search pattern for data that must be matched. If it finds a match, then Oracle Database replaces the data as specified by the `regexp_replace_string` setting. See the following sections for more information:

  – Regular Expression-Based Redaction Policies Using Formats

  – Custom Regular Expression Redaction Policies

- `regexp_replace_string`: Specifies how you want to replace the data to be redacted. See the following sections for more information:

  – Regular Expression-Based Redaction Policies Using Formats

  – Custom Regular Expression Redaction Policies

- `regexp_position`: Specifies the starting position for the string search. The value that you enter must be a positive integer indicating the character of the `column_name` data where Oracle Database should begin the search. The default is `1` or the `DBMS_REDACT.RE_BEGINNING` format, meaning that Oracle Database begins the search at the first character of the `column_name` data.
- `regexp_occurrence`: Specifies how to perform the search and replace operation. The value that you enter must be a non-negative integer indicating the occurrence of the replace operation:

  – If you specify `0` or the `DBMS_REDACT.RE_ALL` format, then Oracle Database replaces all the occurrences of the match.

  - If you specify the `DBMS_REDACT.RE_FIRST` format, then Oracle Database replaces the first occurrence of the match.
  - If you specify a positive integer *n*, then Oracle Database replaces the *n*th occurrence of the match.

  If the occurrence is greater than 1, then the database searches for the second occurrence beginning with the first character following the first occurrence of pattern, and so forth.

- `regexp_match_parameter`: Specifies a text literal that lets you change the default matching behavior of the function. The behavior of this parameter is the same for this function as for the `REGEXP_REPLACE` SQL function. See *Oracle Database SQL Language Reference* for detailed information.

  To filter the search so that it is not case sensitive, specify the `RE_CASE_INSENSITIVE` format.

3.9.3 Regular Expression-Based Redaction Policies Using Formats

The `DBMS_REDACT.ADD_POLICY` procedure supports both `regexp_pattern` and `regexp_replace_string` parameters.

3.9.3.1 Regular Expression Formats

The regular expression formats represent commonly used expressions, such as the replacement of digits within a credit card number.

Table 3-10 describes the formats that you can use with the `regexp_pattern` parameter in the `DBMS_REDACT.ADD_POLICY` procedure.

Table 3-10 Formats for the regexp_pattern Parameter

Format	Description

`DBMS_REDACT.RE_PATTERN_ANY_DIGIT`	Searches for any digit. Replaces the identified pattern with the characters specified by the `regexp_replace_string` parameter. The `DBMS_REDACT.RE_PATTERN_ANY_DIGIT` is commonly used with the following values of the `regexp_replace_string` parameter:
	`regexp_replace_string => DBMS_REDACT.RE_REDACT_WITH_SINGLE_X`
	This setting replaces any matched digit with the `X` character.
	The following setting replaces any matched digit with the `1` character.
	`regexp_replace_string => DBMS_REDACT.RE_REDACT_WITH_SINGLE_1`

Table 3-10 (Cont.) Formats for the regexp_pattern Parameter

Format	Description
`DBMS_REDACT.RE_PATTERN_CC_L6_T4`	Searches for the middle digits of any credit card (other than American Express) that has 6 leading digits and 4 trailing digits. Replaces the identified pattern with the characters specified by the `regexp_replace_string` parameter. The appropriate `regexp_replace_string` setting to use with this format is `DBMS_REDACT.RE_REDACT_CC_MIDDLE_DIGITS`, which finds any credit card that could have 6 leading and 4 trailing digits left as actual data. It then redacts the middle digits.
`DBMS_REDACT.RE_PATTERN_CCN`	Matches credit card numbers other than American Express credit card numbers. The appropriate `regexp_replace_string` setting to use with this format is `DBMS_REDACT.RE_REDACT_CCN`. The end result is a redaction of all the digits except the last 4.
`DBMS_REDACT.RE_PATTERN_AMEX_CCN`	Matches American Express credit card numbers. The appropriate `regexp_replace_string` setting to use with this format is `DBMS_REDACT.RE_REDACT_AMEX_CCN`. The end result is a redaction of all the digits except the last 5.
`DBMS_REDACT.RE_PATTERN_US_PHONE`	Searches for any U.S. telephone number. Replaces the identified pattern with the characters specified by the `regexp_replace_string` parameter The appropriate `regexp_replace_string` setting to use with this format is `DBMS_REDACT.RE_REDACT_US_PHONE_L7`, which finds United States phone numbers and then redacts the last 7 digits.
`DBMS_REDACT.RE_PATTERN_EMAIL_ADDRESS`	Searches for any email address. Replaces the identified pattern with the characters specified by the `regexp_replace_string` parameter The appropriate `regexp_replace_string` settings that you can use with this format are as follows: `RE_REDACT_EMAIL_NAME`, which finds any email address and redacts the email user name `RE_REDACT_EMAIL_DOMAIN`, which finds any email address and redacts the email domain `RE_REDACT_EMAIL_ENTIRE`, which finds any email address and redacts the entire email address
`DBMS_REDACT.RE_PATTERN_IP_ADDRESS`	Searches for an IP address. Replaces the identified pattern with the characters specified by the `regexp_replace_string` parameter. The appropriate `regexp_replace_string` setting to use with this format is `DBMS_REDACT.RE_REDACT_IP_L3`, which replaces the last section of the dotted decimal string representation of an IP address with the characters `999` to indicate that it was redacted.

The table below describes formats that you can use with the `regexp_replace_string` parameter in the `DBMS_REDACT.ADD_POLICY` procedure.

Table 3-11 Formats for the regexp_replace_string Parameter

Format	Description
`DBMS_REDACT.RE_REDACT_WITH_SINGLE_X`	Replaces the data with a single `X` character for each of the actual data characters. For example, the credit card number `5105 1051 0510 5100` could be replaced with `XXXX XXXX XXXX XXXX`.
`DBMS_REDACT.RE_REDACT_WITH_SINGLE_1`	Replaces the data with a single `1` digit for each of the actual data digits. For example, the credit card number `5105 1051 0510 5100` could be replaced with `1111 1111 1111 1111`.
`DBMS_REDACT.RE_REDACT_CC_MIDDLE_DIGITS`	Redacts the middle digits in credit card numbers, as specified by setting the `regexp_pattern` parameter with the `RE_PATTERN_CC_L6_T4` format. The redaction replaces each redacted character with an `X`. For example, the credit card number `5105 1051 0510 5100` could be replaced with `5105 10XX XXXX 5100`.
`DBMS_REDACT.RE_REDACT_CCN`	Redacts the first 12 digits of a credit card number other than an American Express card number. For example, `4012888888881881` is redacted to `************1881`.
`DBMS_REDACT.RE_REDACT_AMEX_CCN`	Redacts the first 10 digits of an American Express number. For example, `378282246310005` is redacted to `**********10005`.
`DBMS_REDACT.RE_REDACT_PHONE_L7`	Redacts the last 7 digits of U.S. telephone numbers, as specified by setting the `regexp_pattern` parameter with the `RE_PATTERN_US_PHONE` format. The redaction replaces each redacted character with an `X`. This setting only applies to hyphenated phone numbers, not phone numbers with spaces. For example, the telephone number `415-555-0100` could be replaced with `415-XXX-XXXX`.
`DBMS_REDACT.RE_REDACT_EMAIL_NAME`	Redacts the email name as specified by setting the `regexp_pattern` parameter with the `RE_PATTERN_EMAIL_ADDRESS` format. The redaction replaces the email user name with four `x` characters. For example, the email address `psmith@example.com` could be replaced with `xxxx@example.com`.
`DBMS_REDACT.RE_REDACT_EMAIL_DOMAIN`	Redacts the email domain name as specified by setting the `regexp_pattern` parameter with the `RE_PATTERN_EMAIL_ADDRESS` format. The redaction replaces the domain with five `x` characters. For example, the email address `psmith@example.com` could be replaced with `psmith@xxxxx.com`.

Table 3-11 (Cont.) Formats for the regexp_replace_string Parameter

Format	Description
`DBMS_REDACT.RE_REDACT_IP_L3`	Redacts the last three digits of the IP address as specified by setting the `regexp_pattern` parameter with the `RE_PATTERN_IP_ADDRESS` format. For example, the IP address `192.0.2.254` could be replaced with `192.0.2.999`, which is an invalid IP address.

Related Topics

- General Syntax of the DBMS_REDACT.ADD_POLICY Procedure
  To create a Data Redaction policy, you must use the `DBMS_REDACT.ADD_POLICY` procedure.

3.9.3.2 Example: Regular Expression Redaction Policy Using Formats

The `DBMS_REDACT.ADD_POLICY` procedure can create a regular expression redaction policy that uses formats.

Example 3-7 shows how to use regular expression formats to redact credit card numbers.

Example 3-7 Regular Expression Data Redaction Character Values

```
BEGIN
 DBMS_REDACT.ADD_POLICY(
   object_schema          => 'mavis',
   object_name            => 'cust_info',
   column_name            => 'cc_num',
   policy_name            => 'redact_cust_cc_nums',
   function_type          => DBMS_REDACT.REGEXP,
   expression             => '1=1',
   regexp_pattern         => DBMS_REDACT.RE_PATTERN_CC_L6_T4,
   regexp_replace_string  => DBMS_REDACT.RE_REDACT_CC_MIDDLE_DIGITS,
   regexp_position        => DBMS_REDACT.RE_BEGINNING,
   regexp_occurrence      => DBMS_REDACT.RE_FIRST,
   regexp_match_parameter => DBMS_REDACT.RE_CASE_INSENSITIVE,
   policy_description     => 'Regular expressions to redact credit card numbers',
   column_description     => 'cc_num contains customer credit card numbers');
END;
/
```

Query and redacted result:

```
SELECT cc_num FROM mavis.cust_info;

CC_NUM
-------
401288XXXXXX1881
411111XXXXXX1111
555555XXXXXX1111
511111XXXXXX1118
```

3.9.4 Custom Regular Expression Redaction Policies

You can customize regular expressions in Data Redaction policies.

3.9.4.1 Settings for Custom Regular Expressions

Oracle Data Redaction provides special settings to configure policies that use regular expressions.

To create custom regular expression redaction policies, you use the following parameters in the `DBMS_REDACT.ADD_POLICY` procedure:

- `regexp_pattern`: This pattern is usually a text literal and can be of any of the data types `CHAR`, `VARCHAR2`, `NCHAR`, or `NVARCHAR2`. The pattern can contain up to 512 bytes. For further information about writing the regular expression for the `regexp_pattern` parameter, see the description of the `pattern` argument of the `REGEXP_REPLACE` SQL function in *Oracle Database SQL Language Reference*, because the support that Data Redaction provides for regular expression matching is similar to that of the `REGEXP_REPLACE` SQL function.
- `regexp_replace_string`: This data can be of any of the data types `CHAR`, `VARCHAR2`, `NCHAR`, or `NVARCHAR2`. The `regexp_replace_string` can contain up to 500 back references to subexpressions in the form `\n`, where *n* is a number from 1 to 9. If you want to include a backslash (\) in the `regexp_replace_string` setting, then you must precede it with the escape character, which is also a backslash. For example, to literally replace the matched pattern with `\2` (rather than replace it with the second matched subexpression of the matched pattern), you enter `\\2` in the `regexp_replace_string` setting. For more information, see *Oracle Database SQL Language Reference*.

See Also:

General Syntax of the DBMS_REDACT.ADD_POLICY Procedure for information about other `DBMS_REDACT.ADD_POLICY` parameters

3.9.4.2 Example: Custom Regular Expression Redaction Policy

The `DBMS_REDACT.ADD_POLICY` procedure `regexp*` parameters can create a custom regular expression redaction policy.

Example 3-8 shows how to use regular expressions to redact the `emp_id` column data. In this example, taken together, the `regexp_pattern` and `regexp_replace_string` parameters do the following: first, find the pattern of 9 digits. For reference, break them into three groups that contain the first 3, the next 2, and then the last 4 digits. Then, replace the first five digits with `XXXXX` concatenated with the third group (the last 4 digits) as found in the original pattern.

Example 3-8 Partially Redacted Data Redaction Using Regular Expressions

```
BEGIN
 DBMS_REDACT.ADD_POLICY(
   object_schema           => 'mavis',
   object_name             => 'cust_info',
   column_name             => 'emp_id',
   policy_name             => 'redact_cust_ids',
   function_type           => DBMS_REDACT.REGEXP,
   expression              => '1=1',
   regexp_pattern          => '(\d\d\d)(\d\d)(\d\d\d\d)',
   regexp_replace_string   => 'XXXXX\3',
   regexp_position         => 1,
   regexp_occurrence       => 0,
   regexp_match_parameter => 'i',
   policy_description     => 'Redacts employee IDs using regular expression',
   column_description     => 'emp_id contains employee ID numbers');
END;
/
```

Query and redacted result:

```
SELECT emp_id FROM mavis.cust_info;

EMP_ID
------------
XXXXX1234
XXXXX5678
```

3.10 Creating a Random Redaction Policy

A random redaction policy presents redacted data as randomly generated values, such as `Ukjsl32[[]]]s` for the character data type.

3.10.1 Syntax for Creating a Random Redaction Policy

A random redaction policy presents the redacted data to the querying application user as randomly generated values, based on the column data type.

Be aware that LOB columns are not supported.

The `DBMS_REDACT.ADD_POLICY` fields for creating a random redaction policy are as follows:

```
DBMS_REDACT.ADD_POLICY (
   object_schema            IN VARCHAR2 := NULL,
   object_name              IN VARCHAR2,
   column_name              IN VARCHAR2 := NULL,
   policy_name              IN VARCHAR2,
   function_type            IN BINARY_INTEGER := NULL,
   expression               IN VARCHAR2,
   enable                   IN BOOLEAN := TRUE);
```

In this specification:

- `object_schema`, `object_name`, `column_name`, `policy_name`, `expression`, `enable`: See General Syntax of the DBMS_REDACT.ADD_POLICY Procedure.
- `function_type`: Specifies the function used to set the type of redaction. Enter `DBMS_REDACT.RANDOM`.

  If you omit the `function_type` parameter, then the default redaction `function_type` setting is `DBMS_REDACT.FULL`.

  Remember that the data type of the column determines which `function_type` settings that you are permitted to use. See Comparison of Full, Partial, Regexp, Random, and Nullify Redaction Based on Data Types.

3.10.2 Example: Random Redaction Policy

You can use the `DBMS_REDACT.ADD_POLICY` PL/SQL procedure to create a random redaction policy.

Example 3-9 shows how to generate random values. Each time you run the `SELECT` statement, the output will be different.

Example 3-9 Randomly Redacted Data Redaction Values

```
BEGIN
 DBMS_REDACT.ADD_POLICY(
   object_schema   => 'mavis',
   object_name     => 'cust_info',
   column_name     => 'login_username',
   policy_name     => 'redact_cust_rand_username',
   function_type   => DBMS_REDACT.RANDOM,
   expression      => 'SYS_CONTEXT(''USERENV'',''SESSION_USER'') = ''APP_USER''');
END;
/
```

Query and redacted result:

```
SELECT login_username FROM mavis.cust_info;

LOGIN_USERNAME
--------------
N[CG{\pTVcK
PL{opergHKV
```

3.11 Creating a Nullify Redaction Policy

You can create Oracle Data Redaction policies that return null values for the displayed value of the table or view column.

3.11.1 About Creating a Policy That Returns Null Values

The `DBMS_REDACT.NULLIFY function_type` parameter redacts all the query-result data in a column and replaces it with null values.

You can use this function type on all supported column data types that the `DBMS_REDACT.NULLIFY` function type supports. It also supports the `CLOB` and `NCLOB` data types. To use the `DBMS_REDACT.NULLIFY` function, you must first ensure that the `COMPATIBLE` parameter is set to `12.2.0.0` or later.

3.11.2 Syntax for Creating a Nullify Redaction Policy

The `DBMS_REDACT.ADD_POLICY` procedure can create a redaction policy that performs a full redaction and displays null values for the redacted columns.

The syntax for using `DBMS_REDACT.ADD_POLICY` to return null values is as follows:

```
DBMS_REDACT.ADD_POLICY (
   object_schema           IN VARCHAR2 := NULL,
   object_name             IN VARCHAR2,
   column_name             IN VARCHAR2 := NULL,
   policy_name             IN VARCHAR2,
   function_type           IN BINARY_INTEGER := NULL,
   expression              IN VARCHAR2,
   enable                  IN BOOLEAN := TRUE);
```

In this specification:

- `object_schema`, `object_name`, `column_name`, `policy_name`, `expression`, `enable`: See General Syntax of the DBMS_REDACT.ADD_POLICY Procedure.
- `function_type`: Specifies the function used to set the type of redaction. Enter `DBMS_REDACT.NULLIFY`.

  If you omit the `function_type` parameter, then the default setting is `DBMS_REDACT.FULL`.

  Remember that the data type of the column determines which `function_type` settings that you are permitted to use. See Comparison of Full, Partial, Regexp, Random, and Nullify Redaction Based on Data Types.

3.11.3 Example: Redaction Policy That Returns Null Values

The `DBMS_REDACT.ADD_POLICY` procedure will add Nullify redaction for the `COMMISSION_PCT` column of the `HR.EMPLOYEES` table.

The `expression` parameter applies the policy to any user who queries the table, except for users who have been granted the `EXEMPT REDACTION POLICY` system or schema privilege.

Example 3-10 shows how to create the Oracle Data Redaction policy.

Example 3-10   Redaction Policy That Returns Null Values

```
BEGIN
 DBMS_REDACT.ADD_POLICY(
   object_schema    => 'hr',
   object_name      => 'employees',
   column_name      => 'commission_pct',
   policy_name      => 'nullify_com_pct',
   function_type    => DBMS_REDACT.NULLIFY,
   expression       => '1=1');
END;
/
```

Query and redacted result:

```
SELECT COMMISSION_PCT FROM HR.EMPLOYEES;

COMMISSION_PCT
--------------
```

Related Topics

- Exemption of Users from Oracle Data Redaction Policies
  You can exempt users from having Oracle Data Redaction policies applied to the data they access.

3.12 Creating a Policy That Uses No Redaction

You can create policies that use no redaction at all, for when you want to test the policy in a development environment.

3.12.1 Syntax for Creating a Policy with No Redaction

The None redaction type option can be used to test the internal operation of redaction policies.

The None redaction type has no effect on the query results against tables that have policies defined on them. You can use this option to test the redaction policy definitions before applying them to a production environment. Be aware that LOB columns are not supported.

The `DBMS_REDACT.ADD_POLICY` fields for creating a policy with no redaction are as follows:

```
DBMS_REDACT.ADD_POLICY (
   object_schema           IN VARCHAR2 := NULL,
   object_name             IN VARCHAR2,
   column_name             IN VARCHAR2 := NULL,
   policy_name             IN VARCHAR2,
   function_type           IN BINARY_INTEGER := NULL,
   expression              IN VARCHAR2,
   enable                  IN BOOLEAN := TRUE);
```

In this specification:

- `object_schema`, `object_name`, `column_name`, `policy_name`, `expression`, `enable`: See General Syntax of the DBMS_REDACT.ADD_POLICY Procedure.
- `function_type`: Specifies the function used to set the type of data redaction. Enter `DBMS_REDACT.NONE`.

  If you omit the `function_type` parameter, then the default redaction `function_type` setting is `DBMS_REDACT.FULL`.

3.12.2 Example: Performing No Redaction

The `DBMS_REDACT.ADD_POLICY` procedure can create a policy that performs no redaction.

Example 3-11 shows how to create a Data Redaction policy that does not redact any of the displayed values.

Example 3-11   No Redacted Data Redaction Values

```
BEGIN
 DBMS_REDACT.ADD_POLICY(
   object_schema      => 'mavis',
   object_name        => 'cust_info',
   column_name        => 'user_name',
   policy_name        => 'redact_cust_no_vals',
   function_type      => DBMS_REDACT.NONE,
   expression         => '1=1');
END;
/
```

Query to get actual data (as data is not redacted) and data output:

```
SELECT user_name FROM mavis.cust_info;

USER_NAME
----------
IDA NEAU
```

3.13 Exemption of Users from Oracle Data Redaction Policies

You can exempt users from having Oracle Data Redaction policies applied to the data they access.

To do so, you should grant the users the `EXEMPT REDACTION POLICY` system or schema privilege. Grant this privilege to trusted users only.

In addition to users who were granted this privilege, user `SYS` is also exempt from all Data Redaction policies. The person who creates the Data Redaction policy is by default not exempt from it, unless this person is user `SYS` or has the `EXEMPT REDACTION POLICY` system or schema privilege.

Note the following:

- The `EXEMPT REDACTION POLICY` system or schema privilege, when granted to a user or role on a schema, enables the grantee to bypass any existing Oracle Data Redaction policies. This user will be able to view the actual data from tables or views on which Data Redaction policies are defined in the schema.
- Users who have the `INSERT` privilege on a table can insert values into a redacted column, regardless of whether a Data Redaction policy exists on the table. Data Redaction only affects SQL `SELECT` statements (that is, queries) issued by a user, and has no effect on any other SQL issued by a user, including `INSERT`, `UPDATE`, `MERGE`, or `DELETE` statements. (See the next bullet for exceptions to this rule.)
- Users cannot perform a `CREATE TABLE AS SELECT` where any of the columns being selected (source columns) is protected by a Data Redaction policy (and similarly, any DML operation like `INSERT-SELECT`, `UPDATE`, `MERGE`, or `DELETE` statements where the source column is a redacted column), unless the user was granted the `EXEMPT REDACTION POLICY` system or schema privilege.
- The `EXEMPT REDACTION POLICY` system privilege is included in the `DBA` role, because the `DBA` role contains the `EXP_FULL_DATABASE` role, which is granted the `EXEMPT REDACTION POLICY` system privilege. This privilege is granted to the `EXP_FULL_DATABASE` role to enable export of data which has one or more redaction policies defined. This privilege must be granted explicitly to users otherwise.

Related Topics

- Restriction of Administrative Access to Oracle Data Redaction Policies
  You can restrict the list of users who can create, view, and edit Data Redaction policies.
- Oracle Data Pump Security Model for Oracle Data Redaction
  The `DATAPUMP_EXP_FULL_DATABASE` role includes the powerful `EXEMPT REDACTION POLICY` system privilege.

3.14 Redacting Multiple Columns

You can redact more than one column in a Data Redaction policy.

3.14.1 Adding Columns to a Data Redaction Policy for a Single Table or View

You can redact columns of different data types, using different redaction types and named policy expressions, for one table or view.

1. Connect to the PDB as a user who has the `EXECUTE` privilege on the `DBMS_REDACT` PL/SQL package and the `ADMINISTER REDACTION POLICY` system or schema privilege.
2. Create the policy for the first column that you want to redact by using `DBMS_REDACT.ADD_POLICY`.
3. Use the `DBMS_REDACT.ALTER_POLICY` procedure to add the next column to the policy.

As necessary, set the action, column_name, function_type, and function_parameters (or the parameters that begin with regexp_) parameters to define the redaction for the new column, but do not change the object_schema, object_name, policy_name, or expression parameters. These parameters remain unchanged for newly redacted columns.

4. Create named policy expressions and apply them to the redacted column.

   If a named policy expression is not applied to the column, the default policy expression from DBMS_REDACT.ADD_POLICY will apply.

Related Topics

- Creating and Managing Multiple Named Policy Expressions
  A named, centrally managed Oracle Data Redaction policy expression can be used in multiple redaction policies and applied to multiple tables or views.

3.14.2 Example: Redacting Multiple Columns

The DBMS_REDACT.ALTER_POLICY procedure can redact multiple columns.

Example 3-12 shows how to add a column to an existing Data Redaction policy. In this example, the action parameter specifies that a new column must be added, using DBMS_REDACT.ADD_COLUMN. The name of the new column, card_num, is set by the column_name parameter.

Example 3-12   Adding a Column to a Data Redaction Policy

```
BEGIN
 DBMS_REDACT.ALTER_POLICY(
  object_schema       => 'mavis',
  object_name         => 'cust_info',
  policy_name         => 'redact_cust_user_ids',
  action              => DBMS_REDACT.ADD_COLUMN,
  column_name         => 'card_num',
  function_type       => DBMS_REDACT.FULL);
END;
/
```

3.15 Altering an Oracle Data Redaction Policy

The DBMS_REDACT.ALTER_POLICY procedure enables you to modify Oracle Data Redaction policies.

3.15.1 About Altering Oracle Data Redaction Policies

The DBMS_REDACT.ALTER_POLICY procedure alters a Data Redaction policy.

If the policy is already enabled, then you do not need to disable it first, and after you alter the policy, it remains enabled.

You can find the names of existing Data Redaction policies by querying the POLICY_NAME column of the REDACTION_POLICIES data dictionary view, and information about the columns, functions, and parameters specified in a policy by querying the REDACTION_COLUMNS view. To find the current value for policies that use full data redaction, you can query the REDACTION_VALUES_FOR_TYPE_FULL data dictionary view.

The `action` parameter specifies the type of modification that you want to perform. At a minimum, you must include the `object_name` and `policy_name` parameters when you run this procedure.

Related Topics

- REDACTION_COLUMNS in the *Oracle Database Reference* guide
- REDACTION_POLICIES in the *Oracle Database Reference* guide
- REDACTION_VALUES_FOR_TYPE_FULL in the *Oracle Database Reference* guide

3.15.2 Syntax for the DBMS_REDACT.ALTER_POLICY Procedure

The `DBMS_REDACT.ALTER_POLICY` procedure syntax can be used to alter all types of Data Redaction policies.

The syntax for the `DBMS_REDACT.ALTER_POLICY` procedure is as follows:

```
DBMS_REDACT.ALTER_POLICY (
   object_schema           IN VARCHAR2 := NULL,
   object_name             IN VARCHAR2,
   policy_name             IN VARCHAR2,
   action                  IN BINARY_INTEGER := DBMS_REDACT.ADD_COLUMN,
   column_name             IN VARCHAR2 := NULL,
   function_type           IN BINARY_INTEGER := DBMS_REDACT.FULL,
   function_parameters     IN VARCHAR2 := NULL,
   expression              IN VARCHAR2 := NULL,
   regexp_pattern          IN VARCHAR2 := NULL,
   regexp_replace_string   IN VARCHAR2 := NULL,
   regexp_position         IN BINARY_INTEGER := 1,
   regexp_occurrence       IN BINARY_INTEGER := 0,
   regexp_match_parameter  IN VARCHAR2 := NULL,
   policy_description      IN VARCHAR2 := NULL,
   column_description      IN VARCHAR2 := NULL);
```

The first time you alter a Data Redaction policy on a column of boolean datatype to turn it into a full redaction policy, which is done by specifying the value `DBMS_REDACT.FULL` for the `function_type` parameter, after the `COMPATIBLE` database initialization parameter is set to 23 or higher, you will see the `BOOLEAN_VALUE` column appear in the catalog view for `REDACTION_VALUES_FOR_TYPE_FULL` and the `BOOLCOL` column appear in the `SYS.RADM_FPTM$` data dictionary table.

In this specification:

- `action`: Enter one of the following values to define the kind of action to use:
  - `DBMS_REDACT.ADD_COLUMN` if you plan to add a new column (in addition to columns that are already protected by the policy) for redaction. This setting is the default for the `action` parameter.
  - `DBMS_REDACT.MODIFY_COLUMN` if you plan to change `function_parameters`, `regexp_*` parameters, or the redaction type in `function_type`.
  - `DBMS_REDACT.DROP_COLUMN` if you want to remove redaction from a column.
  - `DBMS_REDACT.MODIFY_EXPRESSION` if you plan to change the `expression` value. Each policy can have only one policy expression. In other words, when you modify the policy expression, you are replacing the existing policy expression with a new policy expression.

Each column in the table can have a different named policy expression. (You can create and manage multiple named policy expressions.) You can modify named policy expressions by using `DBMS_REDACT.UPDATE_POLICY_EXPRESSION`. You cannot modify named policy expressions by using `DBMS_REDACT.MODIFY_EXPRESSION`.

– `DBMS_REDACT.SET_POLICY_DESCRIPTION` if you want to change the description of the policy.

– `DBMS_REDACT.SET_COLUMN_DESCRIPTION` if you want to change the description of the column.

Related Topics

- Parameters Required for DBMS_REDACT.ALTER_POLICY Actions
  The `DBMS_REDACT.ALTER_POLICY` procedure provides parameters than can perform various actions, such as adding or modifying a column.
- General Syntax of the DBMS_REDACT.ADD_POLICY Procedure
  To create a Data Redaction policy, you must use the `DBMS_REDACT.ADD_POLICY` procedure.
- Creating and Managing Multiple Named Policy Expressions
  A named, centrally managed Oracle Data Redaction policy expression can be used in multiple redaction policies and applied to multiple tables or views.

3.15.3 Parameters Required for DBMS_REDACT.ALTER_POLICY Actions

The `DBMS_REDACT.ALTER_POLICY` procedure provides parameters than can perform various actions, such as adding or modifying a column.

Table 3-12 shows the combinations of these parameters.

Table 3-12 Parameters Required for Various DBMS_REDACT.ALTER_POLICY Actions

Desired Alteration	Parameters to Set
Add a column	• `action` (`DBMS_REDACT.ADD_COLUMN`, optional, default is `DBMS_REDACT.ADD_COLUMN`) • `column_name` (required) • `function_type` (optional, default is `DBMS_REDACT.FULL`) • `function_parameters` (only when `function_type` is `DBMS_REDACT.PARTIAL`) • `regexp`* (only when `function_type` is `DBMS_REDACT.REGEXP` or `DBMS_REDACT.REGEXP_WIDTH`) • `policy_description` (optional) • `column_description` (optional)
Modify a column	• `action` (`DBMS_REDACT.MODIFY_COLUMN`, required) • `column_name` (required) • `function_type` (optional, default is `DBMS_REDACT.FULL`) • `function_parameters` (only when `function_type` is `DBMS_REDACT.PARTIAL`) • `regexp`* (only when `function_type` is `DBMS_REDACT.REGEXP` or `DBMS_REDACT.REGEXP_WIDTH`) • `policy_description` (optional) • `column_description` (optional)
Drop a column	• action (`DBMS_REDACT.DROP_COLUMN`, required) • `column_name` (required)

Table 3-12 (Cont.) Parameters Required for Various DBMS_REDACT.ALTER_POLICY Actions

Desired Alteration	Parameters to Set
Change the policy expression	• `action` (`DBMS_REDACT.MODIFY_EXPRESSION`, required) • `expression` (required) • `policy_description` (optional)
Change the description of the policy	• `action` (`DBMS_REDACT.SET_POLICY_DESCRIPTION`, required) • `policy_description` (required)
Change the description of the column	• `action` (`DBMS_REDACT.SET_COLUMN_DESCRIPTION`, required) • `column_description` (required) • `column_name` (required)

3.15.4 Tutorial: Altering an Oracle Data Redaction Policy

You can redact multiple columns in a table or view, with each column having its own redaction setting and named policy expression.

The exercise in this section shows how to modify a Data Redaction policy so that multiple columns are redacted. It also shows how to change the `expression` setting for the policy. To accomplish this, you must run the `DBMS_REDACT.ALTER_POLICY` procedure in stages.

1. Connect to the PDB as a user who has privileges to create users and grant them privileges.
2. Create the following users:

   ```
   GRANT CREATE SESSION TO dr_admin IDENTIFIED BY password;
   GRANT CREATE SESSION TO sales_rep IDENTIFIED BY password;
   GRANT CREATE SESSION TO support_rep IDENTIFIED BY password;
   ```

3. Grant `EXECUTE` on the `DBMS_REDACT` PL/SQL package to user `dr_admin`.

   ```
   GRANT EXECUTE ON DBMS_REDACT TO dr_admin;
   ```

4. Grant the `ADMINISTER REDACTION POLICY` system privilege to user `dr_admin`.

   ```
   GRANT ADMINISTER REDACTION POLICY TO dr_admin;
   ```

5. Connect as user `OE`.
6. Create and populate a table that contains customer credit card information.

   ```
   CREATE TABLE cust_order_info(
    first_name varchar2(20),
    last_name varchar2(20),
    address varchar2(30),
    city varchar2(30),
    state varchar2(3),
    zip varchar2(5),
    cc_num varchar2(19),
    cc_exp varchar2(7));


   INSERT INTO cust_order_info VALUES ('Jane','Doe','39 Mockingbird Lane', 'San
   Francisco', 'CA', 94114, '5105 1051 0510 5100', '10/2018');
   INSERT INTO cust_order_info VALUES ('Mary','Hightower','2319 Maple Street',
   'Sonoma', 'CA', 95476, '5111 1111 1111 1118', '03/2019');
   ```

```
INSERT INTO cust_order_info VALUES ('Herbert','Donahue','292 Winsome Way', 'San
Francisco', 'CA', 94117, '5454 5454 5454 5454', '08/2018');
```

7. Grant the `SELECT` privilege on the `cust_order_info` table to the `sales_rep` and `support_rep` users.

   ```
   GRANT SELECT ON cust_order_info TO sales_rep, support_rep;
   ```

8. Connect as user `dr_admin`.

9. Modify the policy to include redaction of the expiration date.

   ```
   BEGIN DBMS_REDACT.ALTER_POLICY(
       object_schema      => 'oe',
       object_name        => 'cust_order_info',
       policy_name        => 'redact_cust_cc_info',
       action             => DBMS_REDACT.ADD_COLUMN,
       column_name        => 'cc_exp',
       function_type      => DBMS_REDACT.RANDOM);
   END;
   /
   ```

10. Modify the policy again, to use a condition so that the `sales_rep` user views the redacted values and the `support_rep` user views the actual data.

    ```
    BEGIN
       DBMS_REDACT.ALTER_POLICY(
       object_schema      => 'oe',
       object_name        => 'cust_order_info',
       policy_name        => 'redact_cust_cc_info',
       action             => DBMS_REDACT.MODIFY_EXPRESSION,
       expression         => 'SYS_CONTEXT(''USERENV'',''SESSION_USER'') = ''SALES_REP''');
    END;
    /
    ```

11. To test the policy, have the two users query the `cust_order_info` table.

    First, connect as `support_rep` and query the table.

    ```
    SELECT cc_num, cc_exp FROM OE.cust_order_info;

    CC_NUM               CC_EXP
    -------------------  -------
    5105 1051 0510 5100  10/2018
    5111 1111 1111 1118  03/2019
    5454 5454 5454 5454  08/2018
    ```

    User `support_rep` can view the actual data. Next, connect as `sales_rep` and query the table.

    ```
    SELECT cc_num, cc_exp FROM OE.cust_order_info;

    CC_NUM             CC_EXP
    ----------------   -------
    ************5100   1ST=033
    ************1118   OZA.w4C
    ************5454   B(9+;O1
    ```

    Data is redacted for user `sales_rep`.

12. As user `dr_admin`, alter the `cust_order_info` to include a condition so that only `support_rep` sees the redacted data but `sales_rep` sees the actual data.

    ```
    BEGIN
       DBMS_REDACT.ALTER_POLICY(
    ```

```
   object_schema      => 'oe',
   object_name        => 'cust_order_info',
   policy_name        => 'redact_cust_cc_info',
   action             => DBMS_REDACT.MODIFY_EXPRESSION,
   expression         => 'SYS_CONTEXT(''USERENV'',''SESSION_USER'') =
''SUPPORT_REP''');
END;
/
```

13. Have the users test the policy again.

    First, `support_rep` tests the policy:

```
SELECT cc_num, cc_exp FROM OE.cust_order_info;

CC_NUM              CC_EXP
----------------    -------
************5100    1^XMF~`
************1118    qz+9=#S
************5454    *KCaUkm
```

    User `support_rep` can no longer view the actual data; it is now redacted.

    Next, connect as `sales_rep` and query the table.

```
SELECT cc_num, cc_exp FROM OE.cust_order_info;

CC_NUM                 CC_EXP
-------------------    -------
5105 1051 0510 5100    10/2018
5111 1111 1111 1118    03/2019
5454 5454 5454 5454    08/2018
```

    User `sales_rep` now can view the actual data.

14. If you do not need the components of this tutorial, then you can remove them.

    Connect as `dr_admin` and drop the policy.

```
BEGIN
  DBMS_REDACT.DROP_POLICY (
    object_schema   => 'oe',
    object_name     => 'cust_order_info',
    policy_name     => 'redact_cust_cc_info');
END;
/
```

    Connect as a security administrator and drop the users.

```
DROP USER dr_admin;
DROP USER sales_rep;
DROP USER support_rep;
```

    Connect as user `OE` and drop the `cust_order_info` table.

```
DROP TABLE cust_order_info;
```

3.16 Disabling and Enabling an Oracle Data Redaction Policy

You can disable and then reenable Oracle Data Redactions policies as necessary.

3.16.1 Disabling an Oracle Data Redaction Policy

The `DBMS_REDACT.DISABLE_POLICY` procedure disables Oracle Data Redaction policies.

You can find the names of existing Data Redaction policies and whether they are enabled by querying the `POLICY_NAME` and `ENABLE` columns of the `REDACTION_POLICIES` view. However, as long as the policy still exists, you cannot create another policy for that table or view, even if the original policy is disabled. In other words, if you want to create a different policy on the same column in the same table, then you must drop the first policy before you can create and use the new policy.

1. Connect to the PDB as a user who has the `EXECUTE` privilege on the `DBMS_REDACT` PL/SQL package and the `ADMINISTER REDACTION POLICY` system or schema privilege.
2. Run the `DBMS_REDACT.DISABLE_POLICY` procedure, using the following syntax:

```
DBMS_REDACT.DISABLE_POLICY (
   object_schema       IN VARCHAR2 DEFAULT NULL,
   object_name         IN VARCHAR2,
   policy_name         IN VARCHAR2);
```

   In this specification:

   - `object_schema`: Specifies the schema of the object on which the Data Redaction policy will be disabled. If you omit this setting (or enter `NULL`), then Oracle Database uses the name of the current schema.
   - `object_name`: Specifies the name of the table or view to be used for the Data Redaction policy.
   - `policy_name`: Specifies the name of the policy to be disabled.

   For example:

```
BEGIN
  DBMS_REDACT.DISABLE_POLICY (
    object_schema  => 'mavis',
    object_name    => 'cust_info',
    policy_name    => 'redact_cust_user_ids');
END;
/
```

3.16.2 Enabling an Oracle Data Redaction Policy

The `DBMS_REDACT.ENABLE_POLICY` procedure enables Oracle Data Redaction policies.

Immediately after you create a new policy, you do not need to enable it; the creation process handles that for you. To find the names of existing Data Redaction policies and whether they are enabled, you can query the `POLICY_NAME` and `ENABLE` columns of the `REDACTION_POLICIES` view. After you run the procedure to enable the policy, the enablement takes effect immediately.

1. Connect to the PDB as a user who has the `EXECUTE` privilege on the `DBMS_REDACT` PL/SQL package and the `ADMINISTER REDACTION POLICY` system or schema privilege.
2. Run the `DBMS_REDACT.ENABLE_POLICY` procedure, using the following syntax.

```
DBMS_REDACT.ENABLE_POLICY (
   object_schema       IN VARCHAR2 DEFAULT NULL,
   object_name         IN VARCHAR2,
   policy_name         IN VARCHAR2);
```

In this specification:

- `object_schema`: Specifies the schema of the object on which the Data Redaction policy will be enabled. If you omit this setting (or enter `NULL`), then Oracle Database uses the name of the current schema.
- `object_name`: Specifies the name of the table or view to be used for the Data Redaction policy.
- `policy_name`: Specifies the name of the policy to be enabled.

For example:

```
BEGIN
  DBMS_REDACT.ENABLE_POLICY (
    object_schema  => 'mavis',
    object_name    => 'cust_info',
    policy_name    => 'redact_cust_user_ids');
END;
/
```

3.17 Dropping an Oracle Data Redaction Policy

The `DBMS_REDACT.DROP_POLICY` procedure drops Oracle Data Redaction policies.

You can drop an Oracle Data Redaction policy whether it is enabled or disabled. You can find the names of existing Data Redaction policies, by querying the `POLICY_NAME` column of the `REDACTION_POLICIES` view. When you drop a table or view that is associated with an Oracle Data Redaction policy, the policy is automatically dropped. As a best practice, drop the policy first, and then drop the table or view afterward.

1. Connect to the PDB as a user who has the `EXECUTE` privilege on the `DBMS_REDACT` PL/SQL package and `ADMINISTER REDACTION POLICY` system or schema privilege.
2. Run the `DBMS_REDACT.DROP_POLICY` procedure.

   Use the following syntax:

   ```
   DBMS_REDACT.DROP_POLICY (
      object_schema       IN VARCHAR2 DEFAULT NULL,
      object_name         IN VARCHAR2,
      policy_name         IN VARCHAR2);
   ```

   In this specification:

   - `object_schema`: Specifies the schema of the object to which the Data Redaction policy applies. If you omit this setting (or enter `NULL`), then Oracle Database uses the name of the current schema.
   - `object_name`: Specifies the name of the table or view to be used for the Data Redaction policy.
   - `policy_name`: Specifies the name of the policy to be dropped.

   For example:

   ```
   BEGIN
     DBMS_REDACT.DROP_POLICY (
       object_schema  => 'mavis',
       object_name    => 'cust_info',
       policy_name    => 'redact_cust_user_ids');
   END;
   /
   ```

After you run the DBMS_REDACT.DROP_POLICY procedure, the drop takes effect immediately.

Related Topics

- Dropped Oracle Data Redaction Policies When the Recycle Bin Is Enabled
  You should check if the recycle bin is enabled before you drop Oracle Data Redaction policies.

3.18 Tutorial: SQL Expressions to Build Reports with Redacted Values

SQL expressions can be used to build reports based on columns that have Oracle Data Redaction policies defined on them.

1. Connect to the PDB as a user who has the EXECUTE privilege on the DBMS_REDACT PL/SQL package and the ADMINISTER REDACTION POLICY system or schema privilege.
2. Create the following Data Redaction policy for the HR.EMPLOYEES table.

   This policy will replace the first 4 digits of the value from the SALARY column with the number 9 and the first digit of the value from the COMMISSION_PCT column with a 9.

```
BEGIN
 DBMS_REDACT.ADD_POLICY(
   object_schema          => 'HR',
   object_name            => 'EMPLOYEES',
   column_name            => 'SALARY',
   column_description     => 'Employees salary column shows employee salary',
   policy_name            => 'redact_emp_sal_comm',
   policy_description     => 'Partially redacts the employee salary column',
   function_type          => DBMS_REDACT.PARTIAL,
   function_parameters    => '9,1,4',
   expression             => '1=1');
END;
/
BEGIN
 DBMS_REDACT.ALTER_POLICY(
   object_schema          => 'HR',
   object_name            => 'EMPLOYEES',
   policy_name            => 'redact_emp_sal_comm',
   action                 => DBMS_REDACT.ADD_COLUMN,
   column_name            => 'COMMISSION_PCT',
   function_type          => DBMS_REDACT.PARTIAL,
   function_parameters    => '9,1,1',
   expression             => '1=1');
END;
/
```

3. Connect as the HR user and then run the following report.

   This report will use the SQL expression (SALARY + COMMISSION_PCT) to combine the employees' salaries and commissions.

```
SELECT (SALARY + COMMISSION_PCT) total_emp_compensation
FROM HR.EMPLOYEES
WHERE DEPARTMENT_ID = 80;

TOTAL_EMP_COMPENSATION
----------------------
                0
                0
                0
...
```

4. Use SQL expressions for the report, including concatenation.

   For example:

```
SELECT 'Employee ID '          || EMPLOYEE_ID ||
       ' has a salary of '     || SALARY ||
       ' and a commission of ' || COMMISSION_PCT || '.' detailed_emp_compensation
FROM HR.EMPLOYEES
WHERE DEPARTMENT_ID = 80
ORDER BY EMPLOYEE_ID;

DETAILED_EMP_COMPENSATION
-------------------------------------------------------------
Employee ID 150 has a salary of 99990 and a commission of .95.
Employee ID 151 has a salary of 9999 and a commission of .95.
Employee ID 152 has a salary of 9999 and a commission of .9.
...
```

5. Connect the user who created the `redact_emp_sal_comm` Data Redaction policy.
6. Run the following statement to drop the policy.

```
BEGIN
  DBMS_REDACT.DROP_POLICY (
    object_schema => 'HR',
    object_name   => 'EMPLOYEES',
    policy_name   => 'redact_emp_sal_comm');
END;
/
```

4

Integrating Oracle Data Redaction with Other Oracle Products and Features

Oracle Data Redaction can be used with other Oracle products and database features.

4.1 Oracle Data Redaction and DML and DDL Operations

Oracle Data Redaction affects DML and DDL operations, especially for users who issue SQL against tables with redacted columns.

Note the following:

- If a redacted column appears as the source in a DML operation and the target is not a redacted column, then Oracle Data Redaction considers this as an attempt to circumvent the policy and prevents it with an `ORA-28081: Insufficient privileges. The command references a redacted object` error unless you have the `EXEMPT REDACTION POLICY` privilege.
- If a redacted column appears as the source in a DDL operation, then Oracle Data Redaction considers this as an attempt to circumvent the policy and prevents it with an `ORA-28081: Insufficient privileges. The command references a redacted object` error unless you have the `EXEMPT REDACTION POLICY` privilege.
- Oracle Data Pump issues these kinds of operations, so you may also need to grant the `EXEMPT REDACTION POLICY` privilege, at the schema or system level to a user if they need to perform schema-level exports of tables that have redacted columns. This privilege is included in the `DATABASE_EXP_FULL_DATABASE` role. See Oracle Data Redaction Policy and Object Transfers with Oracle Data Pump for more information.
- Internally, Oracle Data Mining issues DML and DDL operations, so you may need to grant the `EXEMPT REDACTION POLICY` privilege to a user if the user must create data mining models on tables that have redacted columns.

In the following examples, `c1` does not have a redaction policy defined on it while `c2` does.

Example 4-1 DML will result in `ORA-28081` error

```
insert into t1(c1) select c2 from t2;
update t1 set c1 = (select c2 from t2);
delete from t1 where c1 in (select c2 from t2);
```

Example 4-2 DDL will result in `ORA-28081` error

```
create table t3 as select * from t2;
```

Related Topics

- Privileges for Managing Oracle Data Redaction Policies
  Because data redaction involves the protection of highly sensitive data, only trusted users should create Oracle Data Redaction policies.

4.2 Oracle Data Redaction and Nested Functions, Inline Views, and the `WHERE` Clause

Oracle Data Redaction affects nested functions, inline views, and the `WHERE` clause.

- **Nested functions work on actual data** except in the following where they work on redacted data:
  - JSON functions
  - XML functions
  - SQL expressions in cursor expressions
  - Constructor functions
  - PL/SQL functions
  - User defined operators
  - User defined aggregates

  For example:

  ```
  SELECT SUM(AVG(TO_NUMBER ((X)))
  FROM HR.EMPLOYEES
      WHERE .....
  ```

  Although the functions work on the actual data in column `X`, with the execution order being innermost to outermost, the result of the SQL expression gets redacted using full redaction as long as any redacted column is involved in the SQL expression. This is to avoid confusion on which redaction type to use to redact the result of the expression if more than one redacted column is specified in the function and each redacted column has a different function type.
- **Inline views are redacted outermost.** For example, in

  ```
  SELECT XYZ …
      AS SELECT A…
      AS SELECT B…
      AS SELECT C…
  ```

  Only `SELECT XYZ` is redacted after the execution of `SELECT C`, `SELECT B`, and `SELECT A`.
- **The WHERE clause is never redacted.** Oracle Data Redaction redacts data only for the columns in the `SELECT` list.

Related Topics

- Privileges for Managing Oracle Data Redaction Policies
  Because data redaction involves the protection of highly sensitive data, only trusted users should create Oracle Data Redaction policies.

4.3 Redaction Results for Queries with SQL Expressions and Set Operation When Using Columns With Redaction Policies

If a SQL expression is present in the `SELECT` list of query involving columns with redaction policies, the result of the SQL expression will always be fully redacted. The redaction type specified by the user using the `function_type` parameter will not be used to redact the data.

Full redaction is always performed on the result of a SQL expression as it supports the most datatypes.

If the resulting datatype of a SQL expression is not supported by full redaction, `NULL` will be returned as the result.

If SQL expressions or `SET` operations are used on columns with Data Redaction policies, the redaction type for the result will default to `FULL` regardless of the underlying redaction types.

For example, if `c1` has a random redaction type and `c2` has a partial redaction type, then the result of the `SUM` or `UNION` operation will return a fully redacted value.

Example 4-3 `SUM` SQL Expression

```
select sum(c1,c2) from t1 order by c1;
```

Example 4-4 `SET` Operation

```
select c1 from t1 UNION select c2 from t2;
```

Regardless of the individual redaction type used, Data Redaction will fully redact the result of the SQL expression or `SET` operation.

4.4 Oracle Data Redaction Policy and Object Transfers with Oracle Data Pump

Objects with Data Redaction policies can affect Data Pump operations.

4.4.1 Oracle Data Pump Security Model for Oracle Data Redaction

The `DATAPUMP_EXP_FULL_DATABASE` role includes the powerful `EXEMPT REDACTION POLICY` system privilege.

Remember that by default the `DBA` role is granted the `DATAPUMP_EXP_FULL_DATABASE` role as well as the `DATAPUMP_IMP_FULL_DATABASE` role.

This enables users who were granted these roles to be exempt from Data Redaction policies. This means that, when you export objects with Data Redaction policies defined on them, the actual data in the protected tables is copied to the Data Pump target system without being redacted. Users with these roles, including users who were granted the `DBA` role, are able to see the actual data in the target system.

However, by default, all of the Data Redaction policies associated with any tables and views in the Data Pump source system are also included in the export and import operation (along with the objects themselves) and applied to the objects in the target system, so the data is still redacted when users query the objects in the target system.

Related Topics

- Exemption of Users from Oracle Data Redaction Policies
  You can exempt users from having Oracle Data Redaction policies applied to the data they access.

4.4.2 Export and Import of Objects That Have Oracle Data Redaction Policies Defined

You can export objects that have already had Oracle Data Redaction policies defined on them.

4.4.2.1 Finding Object Types Used by Oracle Data Pump

You must find the object types that Oracle Data Pump uses before exporting objects that have Oracle Data Redaction policies defined on them.

After you find these object types, you should use these object types as parameters for the `INCLUDE` directive to the `EXPDP` utility, to selectively export only metadata of these specific object types to the dump file.

- To find the object types, query the `DATABASE_EXPORT_OBJECTS` view.

For example:

```
SELECT OBJECT_PATH
FROM DATABASE_EXPORT_OBJECTS
WHERE OBJECT_PATH LIKE 'RADM_%';
```

Output similar to the following appears:

```
OBJECT_PATH
------------
RADM_FPTM
RADM_POLICY
RADM_POLICY_EXPR
```

4.4.2.2 Exporting Only the Data Dictionary Metadata Related to Data Redaction Policies

You can export only the data dictionary metadata using Oracle Database Pump metadata filters.

This kind of Data Pump export could, for example, be used if you must use the same set of Data Redaction policies and settings across development, test, and production databases. Because the flag `content=metadata_only` is specified, the dump file does not contain any actual data.

- To export only the data dictionary metadata related to data redaction policies, full redaction settings, and policy expressions, include `RADM_POLICY`, `RADM_FPTM`, and `RADM_POLICY_EXPR` in the `EXPDP` utility command similar to the following:

  ```
  expdp system/password \
  full=y \
  COMPRESSION=NONE \
  content=metadata_only \
  INCLUDE=RADM_FPTM,RADM_POLICY,RADM_POLICY_EXPR \
  ```

```
directory=my_directory \
job_name=my_job_name \
dumpfile=my_data_redaction_policy_metadata.dmp
```

Related Topics

- *Oracle Database Utilities*

4.4.2.3 Importing Only the Data Dictionary Metadata Using the INCLUDE Parameter in IMPDP

You can import only the data dictionary metadata using Oracle Database Pump metadata filters.

- To import only the data dictionary metadata related to data redaction policies, full redaction settings, and policy expressions, include `RADM_POLICY`, `RADM_FPTM`, and `RADM_POLICY_EXPR` in the `IMPDP` utility command similar to the following:

  ```
  impdp system/password \
  full=y \
  content=metadata_only \
  INCLUDE=RADM_FPTM,RADM_POLICY,RADM_POLICY_EXPR \
  directory=my_directory \
  job_name=my_job_name \
  dumpfile=my_data_redaction_policy_metadata.dmp
  ```

4.4.3 Export of Data Using the EXPDP Utility access_method Parameter

Oracle Data Pump can export data from a schema that contains an object that has a Data Redaction policy.

If you are using Oracle Data Pump to perform full database export operations using the Data Pump default settings (`direct_path`), and if you receive error messages that you do not understand, then use this section to repeat the operation in such a way as to better understand the error.

If you try to use the Oracle Data Pump Export (`EXPDP`) utility with the `access_method` parameter set to `direct_path` to export data from a schema that contains an object that has a Data Redaction policy defined on it, then the following error message may appear and the export operation fails:

```
ORA-31696: unable to export/import TABLE_DATA:"schema.table" using client specified
DIRECT_PATH method
```

This problem only occurs when you perform a schema-level export as a user who was not granted the `EXP_FULL_DATABASE` role. It does not occur during a full database export, which requires the `EXP_FULL_DATABASE` role. The `EXP_FULL_DATABASE` role includes the `EXEMPT REDACTION POLICY` system privilege, which bypasses Data Redaction policies.

To find the underlying problem, try the `EXPDP` invocation again, but do not set the `access_method` parameter to `direct_path`. Instead, use either `automatic` or `external_table`. The underlying problem could be a permissions problem, for example:

```
ORA-28081: Insufficient privileges - the command references a redacted object.
```

Related Topics

- *Oracle Database Utilities*

4.4.4 Import of Data into Objects Protected by Oracle Data Redaction

During import operations into Oracle Data Redaction-protected objects, be aware of issues that may occur.

Inadvertent Drop of Oracle Data Redaction Policies

During an import operation, be careful that you do not inadvertently drop data redaction policies that protect imported data.

Consider a scenario in which the source tables that were exported using the Oracle Data Pump Export (`EXPDP`) utility do not have Oracle Data Redaction polices. However, the destination tables to which the data is to be imported by using Oracle Data Pump Import (`IMPDP`) have Oracle Data Redaction policies.

During the Data Pump import operation, the status of the Data Redaction policies on the objects being imported depends on the `CONTENT` option of `IMPDP` command.

- If you use the `CONTENT=ALL` or `CONTENT=METADATA_ONLY` option in the `IMPDP` command, then the Data Redaction policies on the destination tables are dropped. You must recreate the Data Redaction policies.
- If you use `CONTENT=DATA_ONLY` in the `IMPDP` command, then the Data Redaction polices on the destination tables are not dropped.

Data Redaction Policy Not Being Imported

During an import operation, you could get an `ORA-28069: A Data Redaction Policy Already Exists On This Object` error.

Consider a scenario in which the source tables that were exported using the Oracle Data Pump Export (`EXPDP`) utility have Oracle Data Redaction policies. The destination tables to which the data is being imported by using Oracle Data Pump (`IMPDP`) also have Oracle Data Redaction policies.

During the Data Pump import operation, you could get the `ORA-28069` error. To avoid this error, you must import only the data using the `CONTENT=DATA_ONLY` parameter. Alternatively, you can drop the redaction policies on the target objects and then import all the data.

Related Topics

- *Oracle Database Utilities*

4.5 Oracle Data Redaction and Oracle Virtual Private Database

Oracle Data Redaction does not affect Oracle Virtual Private Database policies because the VPD inline view, which contains the VPD predicate, acts on actual values. VPD column masking and data redaction policy can be applied on the same columns. VPD column masking is applied first and then the data is redacted.

The main difference between Data Redaction and VPD is that VPD is an access control technology while Data Redaction is used to limit data exposure. You shouldn't use Data Redaction as an access control solution.

Additionally, Oracle Data Redaction differs from Oracle Virtual Private Database in the following ways:

- Oracle Data Redaction provides more redaction features than Oracle Virtual Private Database column masking, which only displays `NULL` for sensitive columns. Many applications cannot support `NULL` values, so Oracle Data Redaction is a good solution for these applications.
- Oracle VPD can provide column and row masking. Data Redaction only affects how the columns' values are displayed, not the rows returned.
- Oracle Virtual Private Database policies can be static, dynamic, and context sensitive, whereas Oracle Data Redaction policies only allow static and context-sensitive policy expressions.
- Oracle Data Redaction permits only one policy to be defined on a table or view, whereas you can define multiple Virtual Private Database policies on an object.
- Data Redaction does not support policies on synonyms, while Oracle VPD does.

Related Topics

- Using Oracle Virtual Private Database to Control Data Access in the *Oracle Database Security Guide*

4.6 Oracle Data Redaction and Oracle Database Real Application Security

The main difference between Data Redaction and Oracle Database Real Application Security is that Real Application Security is an access control technology while Data Redaction is used to limit data exposure. You shouldn't use Data Redaction as an access control solution.

Column security within Real Application Security is based on application privileges that are defined by applications using the Real Application Security framework.

Related Topics

- *Oracle Database Real Application Security Administrator's and Developer's Guide*

4.7 Oracle Data Redaction and Oracle Database Vault

You can use Oracle Data Redaction in an Oracle Database Vault environment.

Oracle Database Vault can be used to limit administrators from disabling, modifying or bypassing Data Redaction policies. Database Vault can protect objects, and their associated Data Redaction policies, from privileged users through realms or command rules.

For example, if there is an Oracle Database Vault realm around an object, a user who does not belong to the authorized list of realm owners or participants cannot see the object data, regardless of whether the user was granted the `EXEMPT REDACTION POLICY` privilege. If the user attempts a DML or DDL statement on the data, error messages result.

Even if there is no realm violation, and if command rules are enabled, then Oracle Database Vault evaluates the associated rule sets. If any of the rule sets evaluate to `FALSE`, then the statement is not allowed to be run, raising a command rule violation.

Related Topics

- *Oracle Database Vault Administrator’s Guide*

4.8 Oracle Data Redaction and Data Masking and Subsetting Pack

Oracle Enterprise Manager Data Masking and Subsetting Pack can be used to create a development or test copy of a production database.

To accomplish this, you can mask this data in bulk, and then put the resulting masked data in the development or test copy.

You can still apply Data Redaction policies to the non-production database, in order to redact columns that contain data that was already masked by Oracle Enterprise Manager Data Masking and Subsetting Pack.

Remember that Oracle Enterprise Manager Data Masking and Subsetting Pack is used to mask data sets when you want to move the data to development and test environments. Data Redaction is mainly designed for redacting at runtime for production applications in a consistent fashion across multiple applications, without having to make application code changes.

Related Topics

- *Oracle Data Masking and Subsetting Guide*

4.9 Oracle Data Redaction and Transparent Data Encryption

Data Redaction and TDE can be used together to encrypt and limit the visibility of your data.

Data Redaction supports TDE tablespace and column level encryption as Data Redaction policies can be defined on encrypted columns.

When retrieving the data, it is first decrypted and then redacted before it is displayed to the user.

Related Topics

- Encrypting Columns in Tables in the *Oracle Database Transparent Data Encryption Guide*

5

Considerations for Oracle Data Redaction

Oracle provides these guidelines for using Oracle Data Redaction securely.

5.1 Oracle Data Redaction General Security Guidelines

It is important to understand the general security guidelines for using Oracle Data Redaction.

- Oracle Data Redaction is not intended to protect against users who run ad hoc SQL queries that attempt to determine the actual values by inference. Inference refers to the process that is designed to find actual data by repeatedly trying queries. For example, to find the users who earn the highest salaries, an intruder could use the following query where column `salary` is redacted:

  ```
  SELECT FIRST_NAME, LAST_NAME, SALARY
  FROM HR.EMPLOYEES
  WHERE SALARY > 16000
  ORDER BY SALARY DESC;

  FIRST_NAME           LAST_NAME                     SALARY
  -------------------- ------------------------- ----------
  Steven               King                               0
  Neena                Kochhar                            0
  Lex                  De Haan                            0
  ```

- Oracle Data Redaction relies on the database and application context values. For applications, it is the responsibility of the application to properly initialize the context value.
- Oracle Data Redaction is not enforced for users who are logged in using the `SYSDBA` administrative privilege.
- Certain DDL statements that attempt to copy the actual data out from under the control of a data redaction policy (that is, `CREATE TABLE AS SELECT`, `INSERT AS SELECT`) are blocked by default, but you can disable this behavior by granting the user the `EXEMPT REDACTION POLICY` privilege.
- Oracle Data Redaction does not affect day-to-day database operations, such as backup and recovery, Oracle Data Pump exports and imports, Oracle Data Guard operations, and replication.

5.1.1 REDACTION_COLUMNS Data Dictionary View Behavior When a View Is Invalid

When an Oracle Data Redaction policy exists on a column of a view, and the view becomes invalid, the Data Redaction policy remains visible in the `REDACTION_COLUMNS` data dictionary view.

For example, a view can become invalid if one of its columns refers to a column that was dropped from a table upon which the view depends.

The column continues to be visible in the REDACTION_COLUMNS data dictionary view because the Data Redaction policy is not automatically dropped when the view becomes invalid.

Instead, the decision on whether to drop the Data Redaction policy is taken when the view is subsequently altered.

This approach was chosen in preference to automatically dropping the Data Redaction policy when the view becomes invalid because it is less error-prone and presents less risk of accidentally displaying actual data from the underlying table.

By deferring the decision to when the view is being altered, it allows the view to be recompiled after the column is restored to the table. After the column is restored to the table and the view is recompiled, then the view becomes valid and still has its Data Redaction policy in place.

On the other hand, if the invalid view definition was subsequently replaced with a valid view definition which no longer contains the column that the Data Redaction policy was previously defined on, it is at that point that the Data Redaction policy is automatically dropped. The REDACTION_COLUMNS data dictionary view is then updated to no longer show the column (since it is no longer part of the new view's definition).

5.1.2 Dropped Oracle Data Redaction Policies When the Recycle Bin Is Enabled

You should check if the recycle bin is enabled before you drop Oracle Data Redaction policies.

If you drop a table or view that has an Oracle Data Redaction policy defined on it when the recycle bin feature is enabled, and if you query the REDACTION_COLUMNS or REDACTION_POLICIES data dictionary views before you purge the recycle bin, then you will see object names such as BIN$... (for example, BIN$1Xu5PSW5VaPgQxGS5AoAEA==$0).

This is normal behavior. These policies are removed when you purge the recycle bin.

To find if the recycle bin is enabled, you can run the SHOW PARAMETER RECYCLEBIN command in SQL*Plus.

Related Topics

- Purging Objects in the Recycle Bin

5.2 Restriction of Administrative Access to Oracle Data Redaction Policies

You can restrict the list of users who can create, view, and edit Data Redaction policies.

To accomplish this, you can limit who has:

- EXECUTE privilege on the DBMS_REDACT package
- ADMINISTER REDACTION POLICY privilege
- SELECT privilege on the REDACTION_POLICIES and REDACTION_COLUMNS views

You can also restrict who is exempted from redaction by limiting the EXEMPT REDACTION POLICY privilege. If you use Oracle Database Vault to restrict privileged user access, you can restrict access to objects with redaction policies. Realms protect the object from being accessed even though the user may have SELECT ANY TABLE privilege.

Related Topics

- Oracle Data Redaction and Oracle Database Vault
  You can use Oracle Data Redaction in an Oracle Database Vault environment.
- Exemption of Users from Oracle Data Redaction Policies
  You can exempt users from having Oracle Data Redaction policies applied to the data they access.
- Introduction to Oracle Database Vault

5.3 How Oracle Data Redaction Affects the SYS, SYSTEM, and Default Schemas

Both users `SYS` and `SYSTEM` automatically have the `EXEMPT REDACTION POLICY` privilege.

`SYSTEM` has the `EXP_FULL_DATABASE` role, which includes the `EXEMPT REDACTION POLICY` privilege.

This means that the `SYS` and `SYSTEM` users can always bypass any existing Oracle Data Redaction policies, and will always be able to view data from tables (or views) that have Data Redaction policies defined on them.

A data redaction policy cannot be applied to an object owned by `SYS`.

Follow these guidelines:

- Do not create Data Redaction policies on the default Oracle Database schemas, including the `SYS` and `SYSTEM` schemas.
- Be aware that granting the `EXEMPT REDACTION POLICY` privilege to additional roles may enable users to bypass Oracle Data Redaction, because the grantee role may have been granted to additional roles.
- Do not revoke the `EXEMPT REDACTION POLICY` privilege from the roles that it was granted to by default.

5.4 Policy Expressions That Use SYS_CONTEXT Attributes

Be careful when writing a policy expression that depends on a `SYS_CONTEXT` attribute that is populated by an application.

If the user somehow connects directly (rather than through the application), then the `SYS_CONTEXT` attribute would not have been populated. If you do not handle this `NULL` scenario in your policy expression, you could unintentionally reveal actual data to the querying user.

For example, suppose you wanted to create a policy expression that intends to redact the query results for everyone except users who have the client identifier value of `SUPERVISOR`. The following expression unintentionally enables querying users who have `NULL` as the value for their `CLIENT_IDENTIFIER` to see the real data:

```
SYS_CONTEXT('USERENV', 'CLIENT_IDENTIFIER') IS NOT 'SUPERVISOR'
```

A more rigorous policy expression redacts the result of the query if the client identifier is not set, that is, it has a `NULL` value.

```
SYS_CONTEXT('USERENV', 'CLIENT_IDENTIFIER') IS NOT 'SUPERVISOR' OR IS NULL
```

Remember that in SQL, comparisons with `NULL` are undefined, and are thus `FALSE`, but redaction only takes place when the policy expression evaluates to `TRUE`.

5.5 Oracle Data Redaction Policies on Materialized Views

You can create Oracle Data Redaction policies on materialized views and on their base tables.

However, ensure that the creator of the materialized view, or the user who performs the refresh of the materialized view, is not blocked by any Data Redaction policies. In other words, the user performing the materialized view creation or refresh operations should be exempt from the Data Redaction policy. When a materialized view is created based on a redacted table, the actual data is copied, but the underlying table's redaction policy is not transferred to the materialized view. To protect the materialized view, a separate redaction policy is required as the underlying table's redaction policy will not apply.

As a best practice, when you create a new materalized view, treat it as a copy of the actual table, and then create a separate Data Redaction policy to protect it.

Oracle Data Warehouse Query Rewrite Operation

Query rewrite may rewrite the query to use a materialized view instead of accessing the detail relations, but only if it can guarantee to deliver exactly the same results as if the rewrite had not occurred. Specifically, the query rewrite must retain and respect any Oracle Data Redaction policies against the relations that are specified in the request query. However, any Oracle Data Redaction policies against the materialized view itself do not have any effect when the materialized view is accessed during query rewrite. This is because the data is already protected by the Oracle Data Redaction policies against the relations in the request query.

5.6 Oracle Data Redaction and Object Dependencies

Objects with dependencies will get recompiled, but you can proactively recompile them yourself.

If the base object that you want to add Data Redaction policies to has dependent objects, then these objects will become invalid when the Data Redaction policy is added to the base object, and these objects will be recompiled automatically when they are used.

Alternatively, you can proactively recompile them yourself by using an `ALTER ... COMPILE` statement. Be aware that invalidating dependent objects (by adding a Data Redaction policy on their base object) and causing them to need to be recompiled can decrease performance in the overall system.

Tip:

Oracle recommends that you only add a Data Redaction policy to an object that has dependent objects during off-peak hours or during a scheduled downtime.

5.7 Oracle Data Redaction and Cursor Expression Queries on Columns Protected by Data Redaction Policies

Cursor expression queries that include the `UNION` clause between SQL statements selecting columns that are protected by Oracle Data Redaction policies may return fewer rows.

This happens because redaction is performed on the column first and then the `UNION` clause is applied on the redacted data from the `SELECT` statements.

To work around this issue, rewrite the queries to include an inline view so that the `UNION` operation occurs before redaction. For example, instead of the following queries:

```
SELECT cursor(SELECT sensitive_column FROM table_name
              UNION
              SELECT sensitive_column FROM table_name);
```

Rewrite the queries as follows:

```
SELECT cursor(SELECT sensitive_column FROM
    (SELECT sensitive_column FROM table_name
    UNION
    SELECT sensitive_column FROM table_name));
```

5.8 Redaction Results of a SQL Expression When Using Multiple Columns With Different Policy Expressions

When a SQL expression uses multiple columns that have different policy expressions applied, the SQL expression is evaluated first and the result of the SQL expression is redacted if the user does not have the `EXEMPT REDACTION POLICY` privilege and if any of the policy expressions evaluates to `TRUE`.

Similarly, if redacted columns are present in the `SELECT` list on either side of set operators such as, `UNION`, `UNION ALL`, `INTERSECT`, `MINUS`, `EXCEPT`, and so on, the result of the SQL expression will be redacted if the user does not have the `EXEMPT REDACTION POLICY` privilege and if any of the policy expressions evaluates to `TRUE`.

For example, in a scenario where column 1 (`c1`) and column 2 (`c2`) have different policy expressions and the following queries are executed, the result will be redacted unless the policy expressions for both column 1 and column 2 evaluate to `FALSE`. This is further described in the table below.

```
SELECT SUM(c1, c2) sum_c1_c2 FROM t1;
SELECT c1 FROM t1 UNION SELECT c2 FROM t1;
```

Table 5-1 Redaction Results of a SQL Expression Based on the Column Policy Expressions

Policy Expression for c1	Policy Expression for c2	SQL Expression Result
`TRUE`	`TRUE`	`redact(sum(c1,c2))`
`FALSE`	`TRUE`	`redact(sum(c1,c2))`
`TRUE`	`FALSE`	`redact(sum(c1,c2))`
`FALSE`	`FALSE`	`sum(c1,c2)`

6

Limitations of Oracle Data Redaction

Oracle Data Redaction can be used with other Oracle features, but some Oracle features may have limitations with regard to Oracle Data Redaction.

6.1 Oracle Data Redaction General Usage Limitations

It is important to understand usage limitations for using Oracle Data Redaction.

- Do not include any redacted columns in the following expressions or functions that are used in a `GROUP BY` clause and `SELECT` list in a SQL statement.
  - JSON operators
  - XML functions
  - SQL expression in a cursor expression
  - Constructor functions
  - User defined operators
  - PL/SQL functions

  Oracle does not support this, and raises an `ORA-00979: not a GROUP BY expression` error. This happens because the expressions or functions in the `SELECT` list must be modified by Data Redaction, but this causes it to no longer be found when it comes time to process the `GROUP BY` clause (which is not updated by Data Redaction) leading to this unintended error message.
- Do not include any redacted columns in the following SQL expressions or functions that are used in both the `DISTINCT` clause and `ORDER BY` clause in a SQL statement.
  - XML functions
  - Cursor expression
  - Constructor functions
  - User defined operators
  - PL/SQL functions

  Oracle does not support this, and raises an error: `ORA-01791: not a SELECTed expression`. This happens because internally the expression in the `SELECT` list must be modified by Data Redaction, but this causes it to no longer be found when it comes time to process the `ORDER BY` clause, leading to this unintended error message.
- Oracle does not support adding a Data Redaction policy on a virtual column:

  While Oracle supports having a Data Redaction policy on the base column of a virtual column, you cannot apply a Data Redaction policy to the virtual column itself.

```
SELECT COLUMN_ID, COLUMN_NAME, VIRTUAL_COLUMN
FROM DBA_TAB_COLS
WHERE TABLE_NAME = 'table_name_that_you_attempted_to_redact'
```

```
AND VIRTUAL_COLUMN = 'YES'
ORDER BY COLUMN_ID;
```

6.2 Oracle Data Redaction and Database Links

Do not create Oracle Data Redaction policies on database views that reference database links.

You can find information about existing database links by querying the `DBA_DB_LINKS` data dictionary view.

Related Topics

- *Oracle Database Administrator's Guide*

6.3 Oracle Data Redaction and Object Types

You can use object types to model real-world entities such as customer accounts.

An object type is a user-defined type. You cannot redact object types. This is because Oracle Database Redaction cannot handle all of the possible ways that object types can be configured, because they are user defined. You can find the type that an object uses by querying the `OBJECT_NAME` and `OBJECT_TYPE` columns of the `ALL_OBJECTS` data dictionary view.

6.4 Oracle Data Redaction and Editions

You cannot redact editioned views. You cannot redact editionable views in an editions-enabled schema.

In addition to these two restrictions, you cannot use a redacted column in the definition of any editiioned view. You can find information about editions by querying the `DBA_EDITIONS` data dictionary view.

6.5 Oracle Data Redaction and JSON

You can use JavaScript Object Notation (JSON) on redacted data.

- You can create `is json` constraints on table columns.

  However, you cannot create an Oracle Data Redaction policy on a table column that has the `is json` constraint. If you attempt to do so, an `ORA-28073 - The column column_name has an unsupported datatype` error is raised. As a workaround solution, Oracle recommends that you create a relational view that uses the `JSON_TABLE` row source operator on top of the JSON object, and then apply the Data Redaction policy to this view.

  The following example shows how you can use `json_table` to create a relational view and add a data redaction policy to this view.

  ```
  create table tab1
          (c1 varchar2(4000) check (c1 is json))

  insert into tab1 values
  ('{"id" : 4,
     "name" : "Large blue soda cup",
     "price" : 12,
     "tags" : ["large1", "green"],
  ```

```
   "phone" : "1-415-555-0100"}'
)

create or replace view json_view1 as
 select t.* from tab1,
  json_table(c1, '$' columns
       id number path '$.id',
       name varchar2(20) path '$.name',
       price number path '$.price',
       phone varchar2(20) path '$.phone') t ;
```

If the owner of the view queries column `phone`, then it is not redacted. Now, the redaction policy can be added on any of the columns in the view (such as `phone`, `price`, and so on).

- You can create JSON duality views on top of redacted data.
  However, there is a restriction to this ability: columns enabled for redaction cannot be primary keys and cannot be part of the Duality View Entity Tag (ETAG).

Related Topics

- *Oracle Database SQL Language Reference*
- *Oracle Database JSON Developer's Guide*
- *JSON-Relational Duality Developer's Guide*

A

Frequently Asked Questions

A.1 Overview Frequently Asked Questions

Find the answers to common questions regarding Oracle Data Redaction

What is Oracle Data Redaction?

Redaction is the process of selectively obscuring sensitive or confidential information that would be returned from a column in a table or view. Data Redaction does not change the contents of the table in the database, it operates on the data displayed when a user performs a query, redacting it based on the policy. Data Redaction can also be referred to as dynamic data masking.

Why do I need Oracle Data Redaction if I already use Transparent Data Encryption?

Oracle Transparent Data Encryption transparently encrypts the data in data files, RMAN backups, and Data Pump exports. The data is transparently *decrypted* when a user or application queries the data. Oracle Data Redaction limits exposure of sensitive data by redacting the column value according to the Data Redaction policy before the result of the query is returned to the user or application.

How does Oracle Data Redaction improve security and improve compliance?

Oracle Data Redaction improves security by dynamically masking sensitive data in real-time, ensuring that unauthorized users cannot access critical information even if they gain access to the database. It allows for granular access control based on user roles and conditions, reducing data exposure and insider threats without altering the stored data. For compliance, it helps organizations adhere to regulations like GDPR, HIPAA, and PCI DSS by auditing the `EXEMPT REDACTION POLICY` privilege and providing you the flexibility to define redaction policies that meet your specific data protection requirements. This reduces the risk of non-compliance, aiding organizations in avoiding fines and maintaining legal standards.

What is the difference between Oracle Data Redaction and tokenization?

Tokenization is a data security technique that replaces sensitive data with a unique, non-sensitive identifier called a token. The token is used instead of the original data, which is stored securely in a separate system. Tokenization usually requires changes to the application code and the application architecture to accommodate the original data being stored in a separate system. This can add complexity and latency to the architecture.

Oracle Data Redaction dynamically redacts sensitive data in real-time at the point of display without altering the actual stored data, using policies to control who can see actual data or only a redacted result. It is non-intrusive and ensures data protection without affecting data integrity. Oracle Data Redaction is built into the kernel of the Oracle Database and requires no application changes and no external software or hardware. Typically, customers will use Data Redaction for read only application screens or web pages, business intelligence or analytic reports, or any application that should redact the result displayed in the report, but will not write the results back to the Oracle Database.

How does Oracle Data Redaction differ from traditional data masking?

Traditional data masking typically involves altering the data stored in the database, while Oracle Data Redaction masks data dynamically at runtime without changing the actual data stored in the database. Oracle Data Redaction is typically used on production databases, where real customer or corporate data resides. Oracle Data Masking and Subsetting is used on non-production databases because it physically changes the data to "production-like" data. Oracle Data Redaction is included, along with Transparent Data Encryption, in the Advanced Security Option. Data masking is included in the Oracle Enterprise Manager Pack, Data Masking and Subsetting, or in the cloud-option Oracle Data Safe.

How does Oracle Data Redaction differ from Oracle Label Security or Virtual Private Database?

Oracle Data Redaction allows you to use full, partial, random, and regular expressions to redact data in your columns, but does not restrict the number of rows returned by a query. Oracle Label Security does not redact column data but restricts the rows returned to a user. Oracle Virtual Private Database (VPD) can limit the column data returned or the rows returned, but can only return null column data. Oracle VPD does not allow partial or regular expression redaction on the column data. Data Redaction, Label Security, and VPD have similar capabilities but fit different use-cases. Oracle encourages you to explore all three options to determine which capability is best suited to your needs.

A.2 Administration Frequently Asked Questions

Find the answers to common questions regarding administration of Oracle Data Redaction.

How do I install Oracle Data Redaction?

Oracle Data Redaction is built into the kernel of the Oracle Database. There are no application changes or intermediary services you need to configure or manage. You create the Data Redaction policies and apply them to the columns in the tables or views you want to redact data for.

What redaction types does Data Redaction provide?

- *Full*: Redacts the entire column value, returning a value of the same data type as the column.
- *Partial*: Redacts specific portions of the data, such as masking parts of a Social Security Number while leaving the last few digits visible.
- *Regular Expression*: Redacts based on pattern matching data within a column. This type is suitable for character data that may vary in length, such as email addresses.
- *Random*: Column data is replaced with random values each time it is accessed, ensuring that the redacted data appears different on each query.
- *Nullify*: Redacts all the data in a column and replaces it with null values.
- *No redaction*: This option enables you to test the internal operations of your redaction policy, with no effect on the results of the queries.

For more information, see Configuring Oracle Data Redaction Policies.

What database privileges are related to Data Redaction?

The privilege `EXEMPT REDACTION POLICY` allows the user to see column data as if there were no redaction policy in place. In Oracle Database 23ai, the `ADMINISTER REDACTION POLICY` privilege allows the granted user the ability to create, modify, or delete Oracle Data Redaction policies.

To create, modify, or delete a Data Redaction policy on an object that they have access to, the user must have the `EXECUTE` privilege on the `DBMS_REDACT` PL/SQL package in addition to the `ADMINISTER REDACTION POLICY` privilege.

How can I stop administrators from disabling, modifying or bypassing Data Redaction policies?

The simplest way is to limit the number of users who have `EXECUTE` privilege on the `DBMS_REDACT` PL/SQL package or the `EXEMPT REDACTION POLICY` privilege. Both users `SYS` and `SYSTEM` automatically have the `EXEMPT REDACTION POLICY` system privilege. `SYSTEM` has the `EXP_FULL_DATABASE` role, which includes the `EXEMPT REDACTION POLICY` system privilege. Both accounts should be used only for critical database operations, such as installation, patching, or upgrading the Oracle Database.

The more robust method to limit administrators from disabling, modifying, or bypassing Data Redaction policies is to use Oracle Database Vault. Oracle Database Vault can protect objects, and their associated Data Redaction policies, from privileged users through realms or command rules. Database Vault is, also, automatically installed in the Oracle Database kernel and available through a separate license.

How complex is it to configure and manage Oracle Data Redaction?

Oracle Data Redaction is easy to configure and manage. Your database administrator or developer can manage the Oracle Data Redaction policies and enable or disable them as necessary.

How do Oracle DBA tasks change with Oracle Data Redaction?

Most DBA tasks remain unchanged with Oracle Data Redaction. Users cannot perform a `CREATE TABLE AS SELECT` where any column being selected (source column) is protected by a Data Redaction policy (and similarly, any DML operation like `INSERT-SELECT`, `UPDATE`, `MERGE`, or `DELETE` statements where the source column is a redacted column), unless the user was granted the `EXEMPT REDACTION POLICY` system or schema privilege.

What is the performance overhead on the database with Oracle Data Redaction?

Oracle Data Redaction minimizes the potential performance impact because it is built into the kernel of the Oracle Database and takes advantage of internal caching and optimization methods. Using a policy expression that always evaluates to true (e.g., `1=1`) can help improve performance because the RDBMS does not have to evaluate the expression.

What types of data can Oracle Data Redaction mask (e.g., DATE, NUMBER, VARCHAR2, CLOB)?

Oracle Data Redaction can perform full redaction on `CHAR`, `VARCHAR2`, `NUMBER`, `FLOAT`, `BINARY_FLOAT`, `BINARY_DOUBLE`, `DATE`, `TIMESTAMP`, `TIMESTAMP WITH TIMEZONE`, `TIMESTAMP WITH LOCAL TIMEZONE`, `BOOLEAN`, `BLOB`, `NCLOB`, and `CLOB` data types and partial, regular expression, or random redaction on a subset of these data types. For more information on Data Redaction's support for Oracle built-in data types and ANSI data types, see Comparison of Full, Partial, Regexp, Random, and Nullify Redaction Based on Data Types.

How do you create a data redaction policy in Oracle?

You can create a Data Redaction policy using the `DBMS_REDACT` PL/SQL package, specifying when redaction should occur through the policy expression parameter.

Then, you apply the Data Redaction policy to specific columns and decide the type of Data Redaction that should take place.

Can redaction be configured to take place only for specific users or roles?

Yes, you can specify which users will see redacted or full data by using the built-in `SYS_CONTEXT` namespace, `USERENV` and either the `SESSION_USER` or `CURRENT_USER`. You can perform the same for roles by using the namespace `SYS_SESSION_ROLES`.

Some administrative users, such as `SYSDBA` or `DBA`, have the `EXEMPT REDACTION POLICY` privilege and are exempt from redaction policies.

How does Oracle Data Redaction integrate with other Oracle security capabilities?

Oracle Data Redaction is a complementary feature to other Oracle Database security capabilities. Oracle Transparent Data Encryption protects data at rest in data files and in Oracle DataPump exports and Oracle RMAN backups. Oracle Database Vault protects privileges and objects but does not support column-level protection or redaction policies. Oracle Data Masking and Subsetting is for use on non-production systems and changes the data to de-identify it from its original values. Oracle Label Security provides row-level controls but no column-level controls. Oracle Virtual Private Database provides both column and row level controls but only supports displaying the value from an unauthorized column as `NULL`. VPD doesn't support displaying part of the actual value, or displaying the result of a regular expression on the actual value. Each feature or option in the Oracle Database security portfolio provides controls to minimize the risk to your data. You will use multiple features/options and some will have overlapping capabilities, which allows you to choose the right capability for your use-cases.

Can Oracle Data Redaction be applied to views and materialized views?

Yes, Oracle Data Redaction can be applied to tables, views, and materialized views.

Can Oracle Data Redaction be managed through Oracle Enterprise Manager?

Yes. Oracle Enterprise Manger Cloud Control provides a management interface for most Oracle Data Redaction features including creating, modifying, or disabling policies and applying or removing policies from columns of table or views.

How do you monitor the performance of Oracle Data Redaction?

Performance can be monitored using Oracle's performance monitoring tools, such as AWR (Automatic Workload Repository) and ADDM (Automatic Database Diagnostic Monitor).

Can I use Oracle Data Redaction with Oracle EBS, PeopleSoft, Siebel, or other packaged applications?

Not directly with the application. You can use Oracle Data Redaction to redact data that is querying *outside* of the application by business intelligence or ad-hoc queries. Most packaged applications offer some type of built-in redaction feature. Please see your packaged application documentation for more information on redacting data within the application.

Can I use Oracle Data Redaction with an in-house developed application?

Yes, it is possible to use Oracle Data Redaction with your in-house developed application, but you must take care to not write the redacted values back to the database. Once data is sent to the client, application, or end user, Oracle does not know whether the data was redacted or not.

B
Using Trace Files to Troubleshoot Oracle Data Redaction Policies

Trace files for Oracle Data Redaction can be generated at either the system level or the session level.

You must set the tracing to `high`. There is no `low` level tracing for Oracle Data Redaction.

1. Connect to the PDB as a user who has the `ALTER SYSTEM` or `ALTER SESSION` privilege.
2. Enable tracing.
   - At the system level, for all sessions:

     ```
     ALTER SYSTEM SET events 'TRACE[RADM] disk=high';
     ```

   - At the session level, for the current session:

     ```
     ALTER SESSION SET events 'TRACE[RADM] disk=high';
     ```

   To find problems with queries involving redacted columns, include `/* KZDMPRQB_DUMP */` in the query. For example:

   ```
   SELECT /* KZDMPRQB_DUMP */ redacted_column FROM
   table_with_redaction_policy;
   ```

   Search for `KCCF2MASK` and `KCCF2MASKDEP` in the dump. The subsequent part of the dump files gives Oracle Redaction-specific insights.
3. To disable tracing:
   - At the system level, for all sessions:

     ```
     ALTER SYSTEM SET EVENTS 'TRACE[RADM] OFF';
     ```

   - At the session level, for the current session:

     ```
     ALTER SESSION SET EVENTS 'TRACE[RADM] OFF';
     ```

C

Oracle Data Redaction Policy Data Dictionary Views

Oracle Database provides data dictionary views that list information about Data Redaction policies.

Before you can query these views, you must be granted the `SELECT_CATALOG_ROLE` role.

C.1 REDACTION_COLUMNS

`REDACTION_COLUMNS` describes all redacted columns in the database, giving the owner of the table or view within which the column resides, the object name, the column name, the type of redaction function, the parameters to the redaction function (if any), and an optional user-provided description of the redaction function that is performed on the column.

Column	Datatype	NULL	ullDescription
`OBJECT_OWNER`	`VARCHAR2(128)`	`NOT NULL`	Owner of the object that is redacted
`OBJECT_NAME`	`VARCHAR2(128)`	`NOT NULL`	Name of the object that is redacted
`COLUMN_NAME`	`VARCHAR2(128)`	`NOT NULL`	Name of the column that is redacted
`FUNCTION_TYPE`	`VARCHAR2(27)`		Redaction function for this column
`FUNCTION_PARAMETERS`	`VARCHAR2(1000)`		Redaction function_parameters for this column
`REGEXP_PATTERN`	`VARCHAR2(512)`		Regular expression pattern to search for
`REGEXP_REPLACE_STRING`	`VARCHAR2(4000)`		Replacement string (up to 4000 characters in length) with up to 500 back-references to subexpressions in the form `\n`, (where `n` is a number from 1 to 9)
`REGEXP_POSITION`	`NUMBER`		Integer counting from 1, giving the position where the search should begin
`REGEXP_OCCURRENCE`	`NUMBER`		Either `0` (to replace all occurrences of the match), or a positive integer `n` (to replace the `n`th occurrence of the match)
`REGEXP_MATCH_PARAMETER`	`VARCHAR2(10)`		To change the default matching behavior, possible values are a combination of `i`, `c`, `n`, `m`, and `x`. See the documentation of the `match_parameter` in the `REGEXP_REPLACE` section of the *Oracle Database SQL Language Reference*.
`COLUMN_DESCRIPTION`	`VARCHAR2(4000)`		User-provided description of the redaction function that is performed on the column. For example, for a Social Security Number column, the description might be: "redact SSN to XXX-XX-(last 4 digits)".

See Also:

Oracle Database Data Redaction Guide for more information about Oracle Data Redaction

C.2 REDACTION_EXPRESSIONS

`REDACTION_EXPRESSIONS` shows all the Data Redaction named Policy Expressions in the database.

Column	Datatype	NULL	Description
`POLICY_EXPRESSION_NAME`	`VARCHAR2(256)`		Customer-specified name of the named Policy Expression
`EXPRESSION`	`VARCHAR2(4000)`		The SQL expression defined for this Data Redaction named Policy Expression
`OBJECT_OWNER`	`VARCHAR2(128)`		Owner of the table or view which this named Policy Expression is associated with
`OBJECT_NAME`	`VARCHAR2(128)`		Name of the table or view which this named Policy Expression is associated with
`COLUMN_NAME`	`VARCHAR2(128)`		Name of the column which this named Policy Expression is associated with
`POLICY_EXPRESSION_DESCRIPTION`	`VARCHAR2(4000)`		Description of this named Policy Expression

C.3 REDACTION_POLICIES

`REDACTION_POLICIES` displays all redaction policies in the database.

Column	Datatype	NULL	Description
`OBJECT_OWNER`	`VARCHAR2(128)`	`NOT NULL`	Owner of the object with the policy
`OBJECT_NAME`	`VARCHAR2(128)`	`NOT NULL`	Name of the object with the policy
`POLICY_NAME`	`VARCHAR2(128)`	`NOT NULL`	Name of the policy
`EXPRESSION`	`VARCHAR2(4000)`	`NOT NULL`	Expression for this policy
`ENABLE`	`VARCHAR2(7)`		Indicates whether the policy is enabled (`YES`) or not (`NO`)
`POLICY_DESCRIPTION`	`VARCHAR2(4000)`		Description of the policy

See Also:

Oracle Database Data Redaction Guide for more information about Oracle Data Redaction

C.4 REDACTION_VALUES_FOR_TYPE_FULL

`REDACTION_VALUES_FOR_TYPE_FULL` shows all of the current values for full redaction.

For example, if a redaction policy is applied to a column of type `BINARY_DOUBLE` and the redaction type is full redaction, that column will be redacted with the value shown in the `BINARY_DOUBLE_VALUE` column of this view.

Column	Datatype	NULL	Description
`NUMBER_VALUE`	`NUMBER`	`NOT NULL`	Redaction result for full redaction on `NUMBER` columns
`BINARY_FLOAT_VALUE`	`BINARY_FLOAT`	`NOT NULL`	Redaction result for full redaction on `BINARY_FLOAT` columns
`BINARY_DOUBLE_VALUE`	`BINARY_DOUBLE`	`NOT NULL`	Redaction result for full redaction on `BINARY_DOUBLE` columns
`CHAR_VALUE`	`VARCHAR2(1)`		Redaction result for full redaction on `CHAR` columns
`VARCHAR_VALUE`	`VARCHAR2(1)`		Redaction result for full redaction on `VARCHAR2` columns
`NCHAR_VALUE`	`NCHAR(1)`		Redaction result for full redaction on `NCHAR` columns
`NVARCHAR_VALUE`	`NVARCHAR2(1)`		Redaction result for full redaction on `NVARCHAR2` columns
`DATE_VALUE`	`DATE`	`NOT NULL`	Redaction result for full redaction on `DATE` columns
`TIMESTAMP_VALUE`	`TIMESTAMP(6)`	`NOT NULL`	Redaction result for full redaction on `TIMESTAMP` columns
`TIMESTAMP_WITH_TIME_ZONE_VALUE`	`TIMESTAMP(6) WITH TIME ZONE`	`NOT NULL`	Redaction result for full redaction on `TIMESTAMP WITH TIME ZONE` columns
`BOOLEAN_VALUE`[1]	`BOOLEAN`	`NOT NULL`	Redaction result for full redaction on `BOOLEAN` columns
`BLOB_VALUE`	`BLOB`		Redaction result for full redaction on `BLOB` columns
`CLOB_VALUE`	`CLOB`		Redaction result for full redaction on `CLOB` columns
`NCLOB_VALUE`	`NCLOB`		Redaction result for full redaction on `NCLOB` columns

1 This column is available starting with Oracle Database 23ai, Release Update 23.7, but it does not appear in this view by default. It is automatically added to this view when the value of the `COMPATIBLE` initialization parameter is `23.0.0` or higher, and a full redaction value for data type `BOOLEAN` is specified or a full redaction policy is defined on any column of `BOOLEAN` data type. See *Oracle Database Data Redaction Guide* for more information.

See Also:

Oracle Database Data Redaction Guide for more information about Oracle Data Redaction

D

DBMS_REDACT

The DBMS_REDACT package provides an interface to Oracle Data Redaction, which enables you to redact data that is returned from queries issued by low-privileged users or an application.

This chapter contains the following topics:

- Overview
- Security Model
- Constants
- Operating Procedures
- Summary of DBMS_REDACT Subprograms

See Also:

- *Oracle Database Data Redaction Guide* regarding using Data Redaction to protect sensitive data

D.1 DBMS_REDACT Overview

Data redaction provides a way to define redaction policies for an application. Oracle Data Redaction provides functionality to redact data that is returned from user SELECT queries in an application. The redaction takes place in real time.

The Data Redaction policy applies to the querying user, depending on this user's SYS_CONTEXT or XS_SYS_CONTEXT values. This redaction process does not require that the queried data be static or unchanging, or for the entire data set to be redacted at one time in an off-line manner. Oracle Database redacts the data only for the rows specified by the user's query, not the data for the entire column. The redaction takes place immediately before the data is returned to the querying user or application.

D.2 DBMS_REDACT Security Model

If the querying user has the EXEMPT REDACTION POLICY privilege, redaction will not be performed. If the user does not have the EXEMPT REDACTION POLICY privilege, the policy expression will be evaluated in the current user's environment. If the policy expression evaluates to TRUE, then redaction will be performed, otherwise no redaction will be performed.

You must have the EXECUTE privilege on the DBMS_REDACT package, as well as the ADMINISTER REDACTION POLICY system or schema privilege, to execute its subprograms. Procedures in the interface are executed with privileges of the current user.

D.3 DBMS_REDACT Constants

The `DBMS_REDACT` package defines several constants for specifying parameter values.

Table D-1 Values for function_type Parameter of DBMS_REDACT.ADD_POLICY

Constant	Value	Type	Description
`FULL`	`1`	`BINARY_INTEGER`	Redact to fixed values
`PARTIAL`	`2`	`BINARY_INTEGER`	Partial redaction, redact a portion of the column data
`RANDOM`	`4`	`BINARY_INTEGER`	Random redaction, each query results in a different random value
`REGEXP`	`5`	`BINARY_INTEGER`	Regular expression based redaction
`REGEXP_WIDTH`	`7`	`BINARY_INTEGER`	Regular expression based redaction that preserves the width of a column that uses a regular expression; designed for applications that use the `OCI_ATTR_CHAR_SIZE` attribute of the Oracle OLE DB Provider interface
`NULLIFY`	`6`	`BINARY_INTEGER`	Returns a null value as a redacted value
`NONE`	`0`	`BINARY_INTEGER`	No redaction

Table D-2 Values for action Parameter of DBMS_REDACT.ALTER_POLICY

Constant	Value	Type	Description
`ADD_COLUMN`	`1`	`BINARY_INTEGER`	Add a column to the redaction policy
`DROP_COLUMN`	`2`	`BINARY_INTEGER`	Drop a column from the redaction policy
`MODIFY_EXPRESSION`	`3`	`BINARY_INTEGER`	Modify the expression of a redaction policy (the expression evaluates to a `BOOLEAN` value: if `TRUE` then redaction is applied, otherwise not)
`MODIFY_COLUMN`	`4`	`BINARY_INTEGER`	Modify a column in the redaction policy to change the redaction `function_type` or the `function_parameters`
`SET_POLICY_DESCRIPTION`	`5`	`BINARY_INTEGER`	Set a description for the redaction policy
`SET_COLUMN_DESCRIPTION`	`6`	`BINARY_INTEGER`	Set a description for the redaction performed on the column

D.4 DBMS_REDACT Operating Procedures

The following table presents the relationship between the type of redaction function and its parameters, based on the datatype of the column being redacted. Examples of the various format strings are provided, showing how to perform some commonplace redaction for a string datatype (in this case, a Social Security Number (SSN)), a `DATE` datatype, and various examples of redaction for the number datatype.

Table D-3 Data Redaction Function Types

function_type	function_parameters	Examples
`DBMS_REDACT.NONE`	-	-
`DBMS_REDACT.FULL`	-	-
`DBMS_REDACT.NULLIFY`	-	-
`DBMS_REDACT.PARTIAL` (for character types)	A comma-separated list, containing the following five fields (with no spaces after the commas delimiting the fields): • `REDACT_PARTIAL_INPUT_FORMAT` • `REDACT_PARTIAL_OUTPUT_FORMAT` • `REDACT_PARTIAL_MASKCHAR` • `REDACT_PARTIAL_MASKFROM` • `REDACT_PARTIAL_MASKTO` See Table D-4.	`'VVVFVVFVVVV,VVV-VV-VVVV,X,1,5'` for redacting the first 5 digits of SSN strings like `123-45-6789`, adding dashes back to format it, resulting in strings like `XXX-XX-6789` `'VVVFVVFVVVV,VVV VV VVVV,X,1,5'` for redacting the first 5 digits of SSN strings like `123-45-6789`, adding spaces to format it, resulting in strings like `XXX XX 6789`
`DBMS_REDACT.PARTIAL` (for character types), continued	`REDACT_PARTIAL_INPUT_FORMAT` - the input format: `V` for value to be possibly redacted, `F` for formatting character to be ignored	The `REDACT_PARTIAL_INPUT_FORMAT` field value `VVVFVVFVVVV` for matching SSN strings like `123-45-6789`
`DBMS_REDACT.PARTIAL` (for character types), continued	`REDACT_PARTIAL_OUTPUT_FORMAT` - the output format: `V` for output of redaction. Any other character will be treated as a formatting character and output literally.	The `REDACT_PARTIAL_OUTPUT_FORMAT` field value `VVV-VV-VVVV` can be used to redact SSN strings into `XXX-XX-6789` (`X` comes from `REDACT_PARTIAL_MASKCHAR` field).
`DBMS_REDACT.PARTIAL` (for character types), continued	`REDACT_PARTIAL_MASKCHAR` - the character used to redact the input	The value `X` for redacting SSN strings into `XXX-XX-6789`.
`DBMS_REDACT.PARTIAL` (for character types), continued	`REDACT_PARTIAL_MASKFROM` - specifies which `V` within the `REDACT_PARTIAL_INPUT_FORMAT` from which to start the redaction (see explanation following the next entry, `REDACT_PARTIAL_MASKTO`)	The value `1` for redacting SSN strings starting at the first `V` of `REDACT_PARTIAL_INPUT_FORMAT` of `VVVFVVFVVVV` into strings like `XXX-XX-6789`

Table D-3 (Cont.) Data Redaction Function Types

function_type	function_parameters	Examples
`DBMS_REDACT.PARTIAL` (for character types), continued	`REDACT_PARTIAL_MASKTO` - specifies which `V` within the `REDACT_PARTIAL_INPUT_FORMAT` at which to end the redaction	The value `5` for redacting SSN strings up to and including the fifth `V` within `REDACT_PARTIAL_INPUT_FORMAT` of `VVVFVVFVVVV` into strings like `XXX-XX-6789`. Note how the format character `'-'` (corresponding to the first `F` within `REDACT_PARTIAL_INPUT_FORMAT`) is ignored as far as redaction is concerned, so the value here is `5` as opposed to `6`.
`DBMS_REDACT.PARTIAL` (for character types), continued	The `REDACT_PARTIAL_MASKFROM` and `REDACT_PARTIAL_MASKTO` field values are specified as counts of the number of `V` characters in the `REDACT_PARTIAL_INPUT_FORMAT` field, up to and including the intended position, starting from the leftmost `V`. This way, `REDACT_PARTIAL_MASKFROM` and `REDACT_PARTIAL_MASKTO` are independent of the specific formatting of the data. For example, in the common use case of redacting an SSN to show only the last four digits, data like `123456789` (with `REDACT_PARTIAL_INPUT_FORMAT` of `VVVVVVVVV`) and data like `123-45-6789` (with `REDACT_PARTIAL_INPUT_FORMAT` of `VVVFVVFVVVV`), would both use `REDACT_PARTIAL_MASKFROM` of `1` and `REDACT_PARTIAL_MASKTO` of `5`.	-
`DBMS_REDACT.PARTIAL` (for number types)	A comma-separated list, containing the following three fields (with no spaces after the commas delimiting the fields): • `REDACT_PARTIAL_MASKCHAR` • `REDACT_PARTIAL_MASKFROM` • `REDACT_PARTIAL_MASKTO` See Table D-4.	`'9,1,5'` for redacting the first 5 digits of an SSN number `123456789` into `999996789`; or `'0,1,2'` for redacting a number `1.23` to `0.03`.

Table D-3 (Cont.) Data Redaction Function Types

function_type	function_parameters	Examples
`DBMS_REDACT.PARTIAL` (for number types), continued	`REDACT_PARTIAL_MASKCHAR` - the character used to redact the input, in the range between `0` and `9` `REDACT_PARTIAL_MASKFROM` - the position, starting from 1, from which to start the redaction. The position does not include the decimal point if it is present. `REDACT_PARTIAL_MASKTO` - the position at which to end the redaction	-
`DBMS_REDACT.PARTIAL` (for datetime datatypes)	A list, containing the following five fields (concatenated so that there is no space between the fields): • `REDACT_PARTIAL_DATE_MONTH` • `REDACT_PARTIAL_DATE_DAY` • `REDACT_PARTIAL_DATE_YEAR` • `REDACT_PARTIAL_DATE_HOUR` • `REDACT_PARTIAL_DATE_MINUTE` • `REDACT_PARTIAL_DATE_SECOND` See Table D-4.	`'m12DYHMS'`, which changes `01-May-01 01:01:01` to `01-Dec-01 01:01:01`.

Table D-3 (Cont.) Data Redaction Function Types

function_type	function_parameters	Examples
DBMS_REDACT.PARTIAL (for datetime datatypes), continued	REDACT_PARTIAL_DATE_MONTH: 'M' (no redacting of month) or 'm#' (redact month to a specific month, if possible), where # (the month specified by its number) is between 1 and 12 REDACT_PARTIAL_DATE_DAY: 'D' (no redacting of date) or 'd#' (redact day to #, if possible), # between 1 and 31 REDACT_PARTIAL_DATE_YEAR: 'Y' (no redacting of year) or 'y#' (redact year to #, if possible), # between 1 and 9999 REDACT_PARTIAL_DATE_HOUR: 'H' (no redacting of hour) or 'h#' (redact hour to #, if possible), # between 0 and 23 REDACT_PARTIAL_DATE_MINUTE: 'M' (no redacting of minute) or 'm#' (redact minute to #, if possible), # between 0 and 59 REDACT_PARTIAL_DATE_SECOND: 'S' (no redacting of second) or 's#' (redact second to #, if possible), # between 0 and 59	-
DBMS_REDACT.REGEXP	-	-
DBMS_REDACT.REGEXP_WIDTH	-	-
DBMS_REDACT.RANDOM	-	-

Table D-4 Format Descriptors with Component Field Names and Delimiters

Datatype	Format Descriptor for Partial redaction
Character	REDACT_PARTIAL_INPUT_FORMAT || ',' || REDACT_PARTIAL_OUTPUT_FORMAT || ',' || REDACT_PARTIAL_MASKCHAR || ',' || REDACT_PARTIAL_MASKFROM || ',' || REDACT_PARTIAL_MASKTO
Number	REDACT_PARTIAL_MASKCHAR || ',' || REDACT_PARTIAL_MASKFROM || ',' || REDACT_PARTIAL_MASKTO
Datetime	REDACT_PARTIAL_DATE_MONTH || REDACT_PARTIAL_DATE_DAY || REDACT_PARTIAL_DATE_YEAR || REDACT_PARTIAL_DATE_HOUR || REDACT_PARTIAL_DATE_MINUTE || REDACT_PARTIAL_DATE_SECOND

D.5 Summary of DBMS_REDACT Subprograms

This table lists and briefly describes the `DBMS_REDACT` package subprograms.

Table D-5 DBMS_REDACT Package Subprograms

Subprogram	Description
ADD_POLICY Procedure	Defines a Data Redaction policy for a table or view
ALTER_POLICY Procedure	Alters a Data Redaction policy for a table or view
APPLY_POLICY_EXPR_TO_COL Procedure	Applies a Data Redaction named policy expression to a redacted column
CREATE_POLICY_EXPRESSION Procedure	Creates a Data Redaction named policy expression
DISABLE_POLICY Procedure	Disables a Data Redaction policy
DROP_POLICY Procedure	Drops a Data Redaction policy
DROP_POLICY_EXPRESSION Procedure	Drops a Data Redaction named policy expression
ENABLE_POLICY Procedure	Enables a Data Redaction policy
UPDATE_FULL_REDACTION_VALUES Procedure	Modifies the default displayed values for a Data Redaction policy for full redaction
UPDATE_POLICY_EXPRESSION Procedure	Updates a Data Redaction named policy expression

D.5.1 ADD_POLICY Procedure

This procedure defines a Data Redaction policy for a table or view.

Syntax

```
DBMS_REDACT.ADD_POLICY (
   object_schema                IN    VARCHAR2 := NULL,
   object_name                  IN    VARCHAR2,
   policy_name                  IN    VARCHAR2,
   column_name                  IN    VARCHAR2 := NULL,
   function_type                IN    BINARY_INTEGER := DBMS_REDACT.FULL,
   function_parameters          IN    VARCHAR2 := NULL,
   expression                   IN    VARCHAR2,
   enable                       IN    BOOLEAN := TRUE,
   regexp_pattern               IN    VARCHAR2 := NULL,
   regexp_replace_string        IN    VARCHAR2 := NULL,
   regexp_position              IN    BINARY_INTEGER := 1,
   regexp_occurrence            IN    BINARY_INTEGER := 0,
   regexp_match_parameter       IN    VARCHAR2 := NULL,
   policy_description           IN    VARCHAR2 := NULL,
   column_description           IN    VARCHAR2 := NULL);
```

Parameters

Table D-6 ***ADD_POLICY Procedure Parameters***

Parameter	Description
`object_schema`	Schema owning the table or view, current user if `NULL`
`object_name`	Name of table or view on which to add a Data Redaction policy
`policy_name`	Name of policy
`column_name`	[Optional] Name of one column to which the redaction policy applies. If you must redact more than one column, use the ALTER_POLICY Procedure to add the additional columns.
`function_type`	Type of redaction function to use. Possible values are: - `DBMS_REDACT.NONE` - `DBMS_REDACT.FULL` (default) - `DBMS_REDACT.NULLIFY` - `DBMS_REDACT.PARTIAL` - `DBMS_REDACT.RANDOM` - `DBMS_REDACT.REGEXP` - `DBMS_REDACT.REGEXP_WIDTH` See Table D-1 for an overview of the meanings of these values, and for some examples of their use.

Table D-6 (Cont.) *ADD_POLICY Procedure Parameters*

Parameter	Description
`function_parameters`	Parameters to the redaction function. The possible values depend on the value of the `function_type` provided. Use `function_parameters` parameter only for `DBMS_REDACT.PARTIAL`. Use `regexp_*` parameters only for `DBMS_REDACT.REGEXP` or `DBMS_REDACT.REGEXP_WIDTH` to define the Data Redaction policy. If the `function_type` is `DBMS_REDACT.REGEXP` or `DBMS_REDACT.REGEXP_WIDTH`, then you must omit the `function_parameters` parameter, and use the `regexp_*` parameters to define the Data Redaction policy. - `DBMS_REDACT.NONE`: Can be omitted entirely and defaults to `NULL` - `DBMS_REDACT.FULL`: Can be omitted entirely and defaults to `NULL` - `DBMS_REDACT.NULLIFY`: Can be omitted entirely and defaults to `NULL` - `DBMS_REDACT.RANDOM`: Can be omitted entirely and defaults to `NULL` - Redacting parameters for partial character redacting. For character datatypes, a comma-separated list containing these fields: • Input format: `'V'` for value to be possibly redacted, `'F'` for formatting character to be ignored • Output format: `'V'` for output of redacting, any other characters will be treated as formatting characters. • Redact character: a character that will be used to replace the actual values. Examples are `'*'` and `'x'`. • Starting digit position: specifies the starting (character) position to begin replacing actual values with the redacting character. The beginning of the string is position 1. Positions do not include formatting characters. • Ending digit position: specifies the ending (character) position to end redacting. An example is `'VVVFVVFVVVV,VVV-VV-VVVV,X,1,5'` for redacting the first 5 digits of SSN string `123-45-6789`, and adding dashes back to format it like an SSN, resulting in `XXX-XX-6789`. For number datatypes, a comma-separated list containing these fields: • Redact character: this is a character between `'0'` to `'9'` that will be used to replace the actual values. • Starting digit position: specifies the starting (digit) position to begin replacing actual values with the redacting character. The beginning of the string is position 1. Positions do not include the decimal point. • Ending digit position: this specifies the ending digit position to end redacting. An example is `'9,1,5'` for redacting the first 5 digits of Social Security Number `123456789`, resulting in `999996789`. For datetime datatypes, the format is a packed string (no spaces or commas) containing the following sequence of fields. Please note that each field can consist of one or more characters, and the field length depends on whether redacting is required. The one-character fields are used to specify that no redaction of that component of the datetime value is to take place. The longer fields indicate a specific

Table D-6 (Cont.) *ADD_POLICY Procedure Parameters*

Parameter	Description
	time or date to use as the redacted value of that component of the datetime value. • Month: `'M'` (no redacting of month) or `'m#'` (redact month to a specific month, if possible), where `#` (the month specified by its number) is between `1` and `12` • Day: `'D'` (no redacting of day) or `'d#'` (redact day to `#`, if possible), `#` between `1` and `31` • Year: `'Y'` (no redacting of year) or `'y#'` (redact year to `#`, if possible), `#` between `1` and `9999` • Hour: `'H'` (no redacting of hour) or `'h#'` (redact hour to `#`, if possible), `#` between `0` and `23` • Minute: `'M'` (no redacting of minute) or `'m#'` (redact minute to `#`, if possible), `#` between `0` and `59` • Second: `'S'` (no redacting of second) or `'s#'` (redact second to `#`, if possible), `#` between `0` and `59` An example is `'m12d1y2001h1m1s1'`, which changes `02-May-13 12:30:23` to `01-Dec-01 01:01:01`. For partial character and number-redacting shortcuts, see *Oracle Database Data Redaction Guide*.
`expression`	Default boolean expression for the table or view. If this expression is used, then redaction takes place only if this policy expression evaluates to `TRUE`. The following functions are supported: • `SYS_CONTEXT` • `XS_SYS_CONTEXT` • `SUBSTR` functions (includes `SUBSTRB`, `SUBSTRC`, `SUBSTR2`, `SUBSTR4`) • `LENGTH` functions (includes `LENGTHB`, `LENGTHC`, `LENGTH2`, `LENGTH4`) • `V` (`APEX_UTIL.GET_SESSION_STATE`) • `NV` (`APEX_UTIL.GET_NUMERIC_SESSION_STATE`) • `OLS_LABEL_DOMINATES` • `DOMINATES` • `OLS_DOMINATES` • `OLS_DOM` • `DOM` • `OLS_STRICTLY_DOMINATES` • `STRICTLY_DOMINATES` • `S_DOM` • `SA_UTL.DOMINATES` • `SA_UTL.CHECK_READ` • `SA_UTL.NUMERIC_LABEL` • `CHAR_TO_LABEL` • `SA_SESSION.LABEL` See *Oracle Database Data Redaction Guide* for more information about these supported functions

Table D-6 (Cont.) *ADD_POLICY Procedure Parameters*

Parameter	Description
`enable`	Boolean value that determines whether the Data Redaction policy is enabled on creation. The default value is `TRUE`, which means that the policy is automatically enabled upon creation. If the `enable` parameter is set to `FALSE`, the policy takes effect only when it is subsequently enabled by calling the `DBMS_REDACT.ENABLE_POLICY` procedure.
`regexp_pattern`	Regular expression pattern up to 512 bytes. Use only if the `function_type` parameter is `DBMS_REDACT.REGEXP` or `DBMS_REDACT.REGEXP_WIDTH`. Also, do not specify the `function_parameters` parameter when `function_type` is `DBMS_REDACT.REGEXP` or `DBMS_REDACT.REGEXP_WIDTH`. See *Oracle Database SQL Language Reference* for more information and examples on using regular expression patterns.
`regexp_replace_string`	Replacement string (up to 4000 characters in length) with up to 500 back-references to subexpressions in the form `\n`, where `n` is a number between 1 and 9. Use only if the `function_type` parameter is `DBMS_REDACT.REGEXP`
`regexp_position`	Integer counting from 1, specifies the position where the search must begin. The default is 1. Use only if the `function_type` parameter is `DBMS_REDACT.REGEXP` or `DBMS_REDACT.REGEXP_WIDTH`.
`regexp_occurrence`	• Use 0 to replace all occurrences of the match. The default is 0. • Use positive integer *`n`* to replace the *`n`*-th occurence of the match. Use only if the `function_type` parameter is `DBMS_REDACT.REGEXP` or `DBMS_REDACT.REGEXP_WIDTH`.
`regexp_match_parameter`	Changes the default matching behavior, possible values are a combination of 'i', 'c', 'n', 'm', 'x' Use only if the `function_type` parameter is `DBMS_REDACT.REGEXP` or `DBMS_REDACT.REGEXP_WIDTH`. See *Oracle Database SQL Language Reference* for more information and examples on using regular expression match parameters.
`policy_description`	Description of redaction policy
`column_description`	Description of the column being redacted

Exceptions

- `ORA-28060` - A Data Redaction policy already exists on this column.
- `ORA-28061` - This object cannot have a Data Redaction policy defined on it.
- `ORA-28062` - The policy expression is too long.
- `ORA-28063` - The policy expression is empty.
- `ORA-28064` - The redaction function is not valid.
- `ORA-28066` - Invalid column *`column`*.
- `ORA-28067` - Missing or invalid column name.

- `ORA-28069` - A Data Redaction policy already exists on this object.
- `ORA-28073` - The column *`column_name`* has an unsupported datatype or attribute.
- `ORA-28074` - The *`field_name`* field of the redaction parameters is not valid.

  The field can be any of the following:
  - `REDACT_PARTIAL_INPUT_FORMAT`
  - `REDACT_PARTIAL_OUTPUT_FORMAT`
  - `REDACT_PARTIAL_MASKCHAR`
  - `REDACT_PARTIAL_MASKFROM`
  - `REDACT_PARTIAL_MASKTO`
  - `REDACT_PARTIAL_DATE_MONTH`
  - `REDACT_PARTIAL_DATE_DAY`
  - `REDACT_PARTIAL_DATE_YEAR`
  - `REDACT_PARTIAL_DATE_HOUR`
  - `REDACT_PARTIAL_DATE_MINUTE`
  - `REDACT_PARTIAL_DATE_SECOND`

  See Table D-3 and Table D-4 for examples of the field contents and field ordering.
- `ORA-28075` - The data redaction policy expression had an error.
- `ORA-28076` - Empty role name specified as `SYS_SESSION_ROLES` attribute.
- `ORA-28077` - The specified role name *`role_name`* for `SYS_SESSION_ROLES` exceeds the maximum length.
- `ORA-28078` - A regular expression parameter is missing or invalid.
- `ORA-28082` - The parameter *parameter* is invalid (where the possible values are `function_parameters`, `column_description`, and `policy_description`).
- `ORA-28085` - The input and output lengths of the redaction do not match.
- `ORA-28086` - The data redaction policy expression had an error
- `ORA-28087` - The policy expression has an unsupported (use of) `operator <operator>`.
- `ORA-28088` - The policy expression has an unsupported PL/SQL function `function_name`.
- `ORA-28089` - Invalid `SYS_CONTEXT` `namespace <namespace>` in policy expression.
- `ORA-28090` - Unsupported nesting of function `function_name` within function `function_name` in policy expression.
- `ORA-28091` - Unsupported use of PL/SQL function `function_name` in policy expression.
- `ORA-28092` - The parameter `parameter_name` with value `value` has an error.
- `ORA-28096` - The parameter `parameter_name` is unexpected for `ADD_POLICY`.
- `ORA-28097` - A data redaction policy cannot be applied to an object owned by `SYS`.
- `ORA-28104` - Input value for *`parameter`* is not valid.

Usage Notes

If you specify `1=1` for the expression parameter, then it is always `TRUE`, which causes the redaction to be performed as long as the querying user is not exempt from the Oracle Data Redaction policy. Specifying `1=1` for the expression parameter will improve Data Redaction performance because Oracle Database will not evaluate the policy expression.

See DBMS_REDACT Security Model for a list of the required privileges for this procedure.

See Operating Procedures for more information regarding function types and function parameters with related examples.

A Data Redaction named policy expression that has been applied to a redacted column takes precedence over the expression defined in the `expression` parameter. To find redacted columns that are affected by named policy expressions, query the `REDACTION_EXPRESSIONS` data dictionary view.

Example

Partial redaction policy:

```
BEGIN
   DBMS_REDACT.ADD_POLICY(
     object_schema        => 'hr',
     object_name          => 'employees',
     column_name          => 'employee_id',
     policy_name          => 'redact_emp_id_nums',
     function_type        => DBMS_REDACT.PARTIAL,
     function_parameters  => '7,1,5',
     expression           => '1=1');
END;
```

Full redaction policy:

```
BEGIN
   DBMS_REDACT.ADD_POLICY(
     object_schema        => 'hr',
     object_name          => 'employees',
     column_name          => 'employee_id',
     policy_name          => 'redact_emp_ids',
     function_type        => DBMS_REDACT.FULL,
     expression           => 'SYS_CONTEXT(''SYS_SESSION_ROLES'',''CLERK'')
                                                                       = ''FALSE''');
END;
```

D.5.2 ALTER_POLICY Procedure

This procedure alters an existing Data Redaction policy for a table or view.

It alters the policy in one or more of the following ways:

- By changing the policy expression
- By changing the type of redaction for a specified column
- By changing the parameters to the redaction function for a specified column
- By adding a column to the redaction policy (the redaction type and any parameters must be specified).
- By removing a column from the redaction policy

- By changing the description of the policy
- By changing the description of the column

Syntax

```
DBMS_REDACT.ALTER_POLICY (
   object_schema                 IN    VARCHAR2 := NULL,
   object_name                   IN    VARCHAR2,
   policy_name                   IN    VARCHAR2,
   action                        IN    BINARY_INTEGER := DBMS_REDACT.ADD_COLUMN,
   column_name                   IN    VARCHAR2 := NULL,
   function_type                 IN    BINARY_INTEGER := DBMS_REDACT.FULL,
   function_parameters           IN    VARCHAR2 := NULL,
   expression                    IN    VARCHAR2,
   regexp_pattern                IN    VARCHAR2 := NULL,
   regexp_replace_string         IN    VARCHAR2 := NULL,
   regexp_position               IN    BINARY_INTEGER := 1,
   regexp_occurrence             IN    BINARY_INTEGER := 0,
   regexp_match_parameter        IN    VARCHAR2 := NULL,
   policy_description            IN    VARCHAR2 := NULL,
   column_description            IN    VARCHAR2 := NULL);
```

Parameters

Table D-7   *ALTER_POLICY Procedure Parameters*

Parameter	Description
`object_schema`	Schema owning the table or view, current user if `NULL`
`object_name`	Name of table or view on which to alter a Data Redaction policy
`policy_name`	Name of policy limited to 30 bytes
`action`	Action to take. For more information see Table D-2.
`column_name`	(Optional) Name of one column to which the redaction policy applies. Required for the following actions: • `add_column` • `drop_column` • `modify_column` • `set_column_description`

Table D-7 (Cont.) *ALTER_POLICY Procedure Parameters*

Parameter	Description
`function_type`	Type of redaction function to use. Possible values are: - `DBMS_REDACT.NONE` - `DBMS_REDACT.FULL` (default) - `DBMS_REDACT.NULLIFY` - `DBMS_REDACT.PARTIAL` - `DBMS_REDACT.RANDOM` - `DBMS_REDACT.REGEXP` - `DBMS_REDACT.REGEXP_WIDTH` If the `function_type` is not `DBMS_REDACT.PARTIAL`, then you must omit the `function_parameters` parameter. If the `function_type` is not `DBMS_REDACT.REGEXP` or `DBMS_REDACT.REGEXP_WIDTH`, then you must omit the `regexp_*` parameters. If the `function_type` is `DBMS_REDACT.REGEXP` or `DBMS_REDACT.REGEXP_WIDTH`, then use the `regexp_pattern`, `regexp_replace_string`, `regexp_position`, `regexp_occurrence`, and `regexp_match_parameter` to define the Data Redaction policy. See Table D-1 for an overview of the meanings of these values, and for some examples of their use.

Table D-7 (Cont.) *ALTER_POLICY Procedure Parameters*

Parameter	Description
`function_parameters`	Parameters to the redaction function. Use only if function_type is `DBMS_REDACT.PARTIAL`. If the `function_type` is `DBMS_REDACT.REGEXP` or `DBMS_REDACT.REGEXP_WIDTH`, then you must omit the `function_parameters` parameter, and use the `regexp_pattern`, `regexp_replace_string`, `regexp_position`, `regexp_occurrence`, and `regexp_match_parameter` to define the Data Redaction policy. - If the `function_type` is `DBMS_REDACT.NONE` or `DBMS_REDACT.NULLIFY`, the `function_parameters` parameter must be omitted. - If the `function_type` is `DBMS_REDACT.FULL` or `DBMS_REDACT.RANDOM`, the `function_parameters` parameter must be omitted. - If the `function_type` is `DBMS_REDACT.PARTIAL`, the `function_parameters` parameter represents the redacting parameters for partial redacting. • Input format: `'V'` for value to be possibly redacted, `'F'` for formatting character to be ignored • Output format: `'V'` for output of redaction, any other characters will be treated as formatting characters • Redact character: a character that will be used to replace the actual values. Examples are `'*'` and `'x'` • Starting digit position: specifies the starting (character) position to begin replacing actual values with the redacting character. The beginning of the string is position 1. Positions do not include formatting characters. • Ending digit position: specifies the ending (character) position to end redacting. An example is `'VVVFVVFVVVV,VVV-VV-VVVV,X,1,5'` for redacting the first 5 digits of SSN string `123-45-6789`, and adding dashes back to format it like an SSN, resulting in `XXX-XX-6789`. For number datatypes, a comma-separated list containing these fields: • Redact character: this is a character between `'0'` to `'9'` that will be used to replace the actual values. • Starting digit position: specifies the starting (digit) position to begin replacing actual values with the redacting character. The beginning of the string is position 1. Positions do not include the decimal point. • Ending digit position: this specifies the ending digit position to end redacting. An example is `'9,1,5'` for redacting the first 5 digits of Social Security Number `123456789`, resulting in `999996789`. For datetime datatypes, the format is a packed string (no spaces or commas) containing the following sequence of fields. Please note that each field can consist of one or more characters, and the field length depends on whether redacting is required. The one-character fields are used to specify that no redaction of that component of the datetime value is to take place. The longer fields indicate a specific time or date to use as the redacted value of that component of the datetime value.

Table D-7 (Cont.) *ALTER_POLICY Procedure Parameters*

Parameter	Description
	• Month: `'M'` (no redacting of month) or `'m#'` (redact month to a specific month, if possible), where `#` (the month specified by its number) is between `1` and `12`. • Day: `'D'` (no redacting of date) or `'d#'` (redact day to `#`, if possible), `#` between `1` and `31`. • Year: `'Y'` (no redacting of year) or `'y#'` (redact year to `#`, if possible), `#` between `1` and `9999`. • Hour: `'H'` (no redacting of hour) or `'h#'` (redact hour to `#`, if possible), `#` between `0` and `23`. • Minute: `'M'` (no redacting of minute) or `'m#'` (redact minute to `#`, if possible), `#` between `0` and `59`. • Second: `'S'` (no redacting of second) or `'s#'` (redact second to `#`, if possible), `#` between `0` and `59`. An example is `'m12DYHMS'`, which changes `01-May-01 01:01:01` to `01-Dec-01 01:01:01` For partial character and number-redacting shortcuts, see *Oracle Database Data Redaction Guide*.
`expression`	Default boolean expression for the table or view. Redaction takes place only if this policy expression evaluates to `TRUE`. The following functions are supported: • `SYS_CONTEXT` • `XS_SYS_CONTEXT` (See *Oracle Database Data Redaction Guide* • `NV` (`APEX_UTIL.GET_NUMERIC_SESSION_STATE`) • `V` (`APEX_UTIL.GET_SESSION_STATE`) • `OLS_LABEL_DOMINATES` • `DOMINATES` • `OLS_DOMINATES` • `OLS_DOM` • `DOM` • `OLS_STRICTLY_DOMINATES` • `STRICTLY_DOMINATES` • `S_DOM` • `SA_UTL.DOMINATES` • `SA_UTL.CHECK_READ` • `SA_UTL.NUMERIC_LABEL` • `CHAR_TO_LABEL` • `SA_SESSION.LABEL`
`regexp_pattern`	Regular expression pattern up to 512 bytes. Use only if the `function_type` parameter is `DBMS_REDACT.REGEXP` or `DBMS_REDACT.REGEXP_WIDTH`. Also, do not specify the `function_parameters` parameter when `function_type` is `DBMS_REDACT.REGEXP` or `DBMS_REDACT.REGEXP_WIDTH`. See *Oracle Database SQL Language Reference* for more information and examples on using regular expression patterns.

Table D-7 (Cont.) *ALTER_POLICY Procedure Parameters*

Parameter	Description
`regexp_replace_string`	Replacement string (up to 4000 characters in length) with up to 500 back-references to subexpressions in the form `\n`, where `n` is a number between 1 and 9. Use only if the `function_type` parameter is `DBMS_REDACT.REGEXP` or `DBMS_REDACT.REGEXP_WIDTH`.
`regexp_position`	Integer counting from 1, specifies the position where the search must begin. Use only if the `function_type` parameter is `DBMS_REDACT.REGEXP` or `DBMS_REDACT.REGEXP_WIDTH`. The default is 1 meaning that Oracle Database begins the search at the first character of the `column_name` data.
`regexp_occurrence`	• Use 0 to replace all occurrences of the match. 0 is the default. • Use positive integer *`n`* to replace the *`n`*-th occurence of the match. Use only if the `function_type` parameter is `DBMS_REDACT.REGEXP` or `DBMS_REDACT.REGEXP_WIDTH`.
`regexp_match_parameter`	Changes the default matching behavior, possible values are a combination of 'i', 'c', 'ic', 'iq', 'n', 'm', 'x'. The behavior of this parameter is the same for this function as for the `REGEXP_REPLACE` SQL function. Use only if the `function_type` parameter is `DBMS_REDACT.REGEXP` or `DBMS_REDACT.REGEXP_WIDTH`. See *Oracle Database SQL Language Reference* for more information and examples on using regular expression match parameters.
`policy_description`	Description of redaction policy
`column_description`	Description of the column being redacted

Exceptions

- `ORA-28060` - A data redaction policy already exists on this column.
- `ORA-28061` - This object cannot have a data redaction policy defined on it.
- `ORA-28062` - The policy expression is too long.
- `ORA-28063` - The policy expression is empty.
- `ORA-28064` - The redaction function is not valid.
- `ORA-28066` - Invalid column *`column`*.
- `ORA-28067` - Missing or invalid column name.
- `ORA-28068` - The object *`object`* does not have a Data Redaction policy.
- `ORA-28069` - A Data Redaction policy already exists on this object.
- `ORA-28070` - The column *`column`* does not have a Data Redaction policy.
- `ORA-28071` - The action is not valid.
- `ORA-28072` - The specified policy name is incorrect.
- `ORA-28073` - The column *`column_name`* has an unsupported datatype or attribute.
- `ORA-28074` - The *`field_name`* field of the redaction parameters is not valid.

The field can be any of the following:

- `REDACT_PARTIAL_INPUT_FORMAT`
- `REDACT_PARTIAL_OUTPUT_FORMAT`
- `REDACT_PARTIAL_MASKCHAR`
- `REDACT_PARTIAL_MASKFROM`
- `REDACT_PARTIAL_MASKTO`
- `REDACT_PARTIAL_DATE_MONTH`
- `REDACT_PARTIAL_DATE_DAY`
- `REDACT_PARTIAL_DATE_YEAR`
- `REDACT_PARTIAL_DATE_HOUR`
- `REDACT_PARTIAL_DATE_MINUTE`
- `REDACT_PARTIAL_DATE_SECOND`

See Table D-3 and Table D-4 for examples of the field contents and field ordering.

- `ORA-28075` - The data redaction policy expression had an error.
- `ORA-28076` - Empty role name specified as `SYS_SESSION_ROLES` attribute.
- `ORA-28077` - The specified role name *`role_name`* for `SYS_SESSION_ROLES` exceeds the maximum length.
- `ORA-28078` - A regular expression parameter is missing or invalid.
- `ORA-28082` - The parameter *parameter* is invalid (where the possible values are `function_parameters`, `column_description`, and `policy_description`).
- `ORA-28085` - The input and output lengths of the redaction do not match.
- `ORA-28086` - The data redaction policy expression had an error
- `ORA-28087` - The policy expression has an unsupported (use of) `operator <operator>`.
- `ORA-28088` - The policy expression has an unsupported PL/SQL function `function_name`.
- `ORA-28089` - Invalid `SYS_CONTEXT` `namespace <namespace>` in policy expression.
- `ORA-28090` - Unsupported nesting of function `function_name` within function `function_name` in policy expression.
- `ORA-28091` - Unsupported use of PL/SQL function `function_name` in policy expression.
- `ORA-28092` - The parameter `parameter_name` with value `value` has an error.
- `ORA-28095` - The parameter `parameter_name` is unexpected for `ACTION <action>` for `ALTER_POLICTY`.
- `ORA-28097` - A data redaction policy cannot be applied to an object owned by `SYS`.
- `ORA-28104` - Input value for *`parameter`* is not valid.

Usage Notes

See DBMS_REDACT Security Model for a list of the required privileges for this procedure.

See Operating Procedures for more information regarding Function Types and Function Parameters with related examples.

A Data Redaction named policy expression that has been applied to a redacted column takes precedence over the expression defined in the `expression` parameter. To find redacted columns that are affected by named policy expressions, query the `REDACTION_EXPRESSIONS` data dictionary view.

Examples

```
BEGIN
   DBMS_REDACT.ALTER_POLICY(
     object_schema          => 'HR',
     object_name            => 'EMPLOYEES',
     policy_name            => 'redact_emp_id_nums',
     action                 => DBMS_REDACT.DROP_COLUMN,
     column_name            => 'EMAIL');
END;
```

D.5.3 APPLY_POLICY_EXPR_TO_COL Procedure

This procedure associates an Oracle Data Redaction named policy expression with a redacted column from a table or view.

Syntax

```
DBMS_REDACT.APPLY_POLICY_EXPR_TO_COL (
   object_schema                IN VARCHAR2 := NULL,
   object_name                  IN VARCHAR2,
   column_name                  IN VARCHAR2,
   policy_expression_name       IN VARCHAR2 := NULL);
```

Parameters

Table D-8 ***APPLY_POLICY_EXPR_TO_COL Procedure Parameters***

Parameter	Description
`object_schema`	Name of the schema that contains the redacted column. If omitted, then the current schema is used.
`object_name`	Name of the object (table or view) that contains the redacted column
`column_name`	Name of the redacted column to which the policy expression is applied
`policy_expression_name`	If `NULL`, then the named policy expression associated with the redacted column `column_name` is removed.

Exceptions

- `ORA-28061` - This object cannot have a data redaction policy defined on it.
- `ORA-28066` - Invalid column `column_name`.
- `ORA-28067` - Missing or invalid column name.
- `ORA-28068` - The object *object* does not have a Data Redaction policy.
- `ORA-28070` - The column `column_name` does not have a Data Redaction policy.
- `ORA-28082` - The parameter *parameter* is invalid.
- `ORA-28092` - The parameter *parameter* with value *value* has an error.

Usage Notes

See DBMS_REDACT Security Model for a list of the required privileges for this procedure.

You can find existing Data Redaction policy expressions by querying the `REDACTION_EXPRESSIONS` data dictionary view. To find columns that have been redacted, query the `REDACTION_COLUMNS` data dictionary view.

Example

```
BEGIN
   DBMS_REDACT.APPLY_POLICY_EXPR_TO_COL(
      object_schema            => 'OE',
      object_name              => 'CUSTOMERS',
      column_name              => 'INCOME_LEVEL',
      policy_expression_name   => 'oe_redact_pol');
END;
```

D.5.4 CREATE_POLICY_EXPRESSION Procedure

This procedure creates an Oracle Data Redaction named policy expression.

Syntax

```
DBMS_REDACT.CREATE_POLICY_EXPRESSION (
   policy_expression_name          IN    VARCHAR2,
   expression                      IN    VARCHAR2,
   policy_expression_description   IN    VARCHAR2 := NULL);
```

Parameters

Table D-9 ***CREATE_POLICY_EXPRESSION Procedure Parameters***

Parameter	Description
`policy_expression_name`	Name of the policy expression
`expression`	Definition of the policy expression.
`policy_expression_description`	Description of the policy expression

Exceptions

- `ORA-28082` - The parameter *parameter* is invalid.
- `ORA-28092` - The parameter *parameter* with value *value* has an error.

Usage Notes

If the expression parameter is `1=1`, then it is always `TRUE`, which causes the redaction to be performed as long as the querying user is not exempt from the Oracle Data Redaction policy. Specifying this parameter as `1=1` will improve Data Redaction performance because Oracle Database will not evaluate the policy expression.

See DBMS_REDACT Security Model for a list of the required privileges for this procedure.

See APPLY_POLICY_EXPR_TO_COL Procedure for how to apply policy expressions to table columns.

After you create a policy expression, you can associate it with a redacted table or view column by running the `DBMS_REDACT.APPLY_POLICY_EXPR_TO_COL` procedure. To find existing redacted columns, query the `REDACTION_COLUMNS` data dictionary view.

Example

```
BEGIN
   DBMS_REDACT.CREATE_POLICY_EXPRESSION(
     policy_expression_name          => 'oe_redact_pol',
     expression                      => 'SYS_CONTEXT(''USERENV'',''SESSION_USER'') =
''OE'''),
     policy_expression_description   => 'Enables policy for user OE ');
END;
```

D.5.5 DISABLE_POLICY Procedure

This procedure disables a Data Redaction policy.

Syntax

```
DBMS_REDACT.DISABLE_POLICY (
   object_schema                  IN     VARCHAR2 := NULL,
   object_name                    IN     VARCHAR2,
   policy_name                    IN     VARCHAR2);
```

Parameters

Table D-10 *DISABLE_POLICY Procedure Parameters*

Parameter	Description
`object_schema`	Schema owning the table or view, current user if `NULL`
`object_name`	Name of table or view for which to disable a Data Redaction policy
`policy_name`	Name of policy to be disabled

Exceptions

- `ORA_28061` - This object cannot have a Data Redaction policy defined on it.
- `ORA-28068` - The object *`object`* does not have a Data Redaction policy.
- `ORA-28072` - The specified policy name is incorrect.
- `ORA-28080` - The policy was already disabled.
- `ORA_28104` - Input value for `policy-name` is not valid.

Usage Notes

See DBMS_REDACT Security Model for a list of the required privileges for this procedure.

Examples

```
BEGIN
   DBMS_REDACT.DISABLE_POLICY (
     object_schema     => 'hr',
     object_name       => 'employees',
     policy_name       => 'redact_emp_ids');
END;
```

D.5.6 DROP_POLICY Procedure

This procedure drops a Data Redaction policy by removing a redaction policy from the table or view.

Syntax

```
DBMS_REDACT.DROP_POLICY (
   object_schema                IN    VARCHAR2 := NULL,
   object_name                  IN    VARCHAR2,
   policy_name                  IN    VARCHAR2);
```

Parameters

Table D-11   *DROP_POLICY Procedure Parameters*

Parameter	Description
`object_schema`	Schema owning the table or view, current user if `NULL`
`object_name`	Name of table or view from which to drop a Data Redaction policy
`policy_name`	Name of policy to be dropped

Exceptions

- `ORA_28061` - This object cannot have a Data Redaction policy defined on it.
- `ORA-28068` - The object *`object`* does not have a Data Redaction policy.
- `ORA-28072` - The specified policy name is incorrect.
- `ORA_28104` - Input value for `policy-name` is not valid.

Usage Notes

See DBMS_REDACT Security Model for a list of the required privileges for this procedure.

Examples

```
BEGIN
   DBMS_REDACT.DROP_POLICY (
      object_schema     =>  'hr',
      object_name       =>  'employees',
      policy_name       =>  'redact_emp_ids');
END;
```

D.5.7 DROP_POLICY_EXPRESSION Procedure

This procedure drops a named policy expression.

Syntax

```
DBMS_REDACT.DROP_POLICY_EXPRESSION (
   policy_expression_name       IN    VARCHAR2);
```

Parameters

Table D-12 *DROP_POLICY_EXPRESSION Procedure Parameters*

Parameter	Description
`policy_expression_name`	Name of the policy expression

Exceptions

- `ORA-28082` - The parameter *parameter* is invalid.
- `ORA-28092` - The parameter *parameter* with value *value* has an error.

Usage Notes

See DBMS_REDACT Security Model for a list of the required privileges for this procedure.

You can find existing Data Redaction policy expressions by querying the `REDACTION_EXPRESSIONS` data dictionary view.

Example

```
BEGIN
   DBMS_REDACT.DROP_POLICY_EXPRESSION(
     policy_expression_name  => 'oe_redact_pol');
END;
```

D.5.8 ENABLE_POLICY Procedure

This procedure re-enables a Data Redaction policy.

Syntax

```
DBMS_REDACT.ENABLE_POLICY (
   object_schema                 IN     VARCHAR2 := NULL,
   object_name                   IN     VARCHAR2,
   policy_name                   IN     VARCHAR2);
```

Parameters

Table D-13 *ENABLE_POLICY Procedure Parameters*

Parameter	Description
`object_schema`	Schema owning the table or view, current user if `NULL`
`object_name`	Name of table or view on which to enable a Data Redaction policy
`policy_name`	Name of policy to be enabled

Exceptions

- `ORA-28068` - The object *`object`* does not have a Data Redaction policy.
- `ORA-28071` - The action is not valid.
- `ORA-28072` - The specified policy name is incorrect.
- `ORA-28079` - The policy was already enabled.

Usage Notes

See DBMS_REDACT Security Model for a list of the required privileges for this procedure.

Examples

```
BEGIN
   DBMS_REDACT.ENABLE_POLICY (
      object_schema    =>  'hr',
      object_name      =>  'employees',
      policy_name      =>  'redact_emp_ids');
END;
```

D.5.9 UPDATE_FULL_REDACTION_VALUES Procedure

This procedure modifies the redacted output that will be displayed when querying a column having a Data Redaction policy where the `function_type` is `DBMS_REDACT.FULL`. Restart the database instance to apply these changes.

Syntax

```
DBMS_REDACT.UPDATE_FULL_REDACTION_VALUES (
   number_val       IN NUMBER                     := NULL,
   binfloat_val     IN BINARY_FLOAT               := NULL,
   bindouble_val    IN BINARY_DOUBLE              := NULL,
   char_val         IN CHAR                       := NULL,
   varchar_val      IN VARCHAR2                   := NULL,
   nchar_val        IN NCHAR                      := NULL,
   nvarchar_val     IN NVARCHAR2                  := NULL,
   date_val         IN DATE                       := NULL,
   ts_val           IN TIMESTAMP                  := NULL,
   tswtz_val        IN TIMESTAMP WITH TIME ZONE   := NULL,
   blob_val         IN BLOB                       := NULL,
   clob_val         IN CLOB                       := NULL,
   nclob_val        IN NCLOB                      := NULL,
   boolean_val      IN BOOLEAN                    := NULL);
```

Parameters

Table D-14 ***UPDATE_FULL_REDACTION_VALUES Procedure Parameters***

Parameter	Description
`number_val`	Modifies the default value for columns of the `NUMBER` datatype
`binfloat_val`	Modifies the default value for columns of the `BINARY_FLOAT` datatype
`bindouble_val`	Modifies the default value for columns of the `BINARY_DOUBLE` datatype
`char_val`	Modifies the default value for columns of the `CHAR` datatype
`varchar_val`	Modifies the default value for columns of the `VARCHAR2` datatype
`nchar_val`	Modifies the default value for columns of the `NCHAR` datatype
`nvarchar_val`	Modifies the default value for columns of the `NVARCHAR2` datatype
`date`	Modifies the default value for columns of the `DATE` datatype
`ts_val`	Modifies the default value for columns of the `TIMESTAMP` datatype
`tswtz_val`	Modifies the default value for columns of the `TIMESTAMP WITH TIME ZONE` datatype

Table D-14 (Cont.) *UPDATE_FULL_REDACTION_VALUES Procedure Parameters*

Parameter	Description
`blob_val`	Modifies the default value for columns of the `BLOB` datatype
`clob_val`	Modifies the default value for columns of the `CLOB` datatype
`nclob_val`	Modifies the default value for columns of the `NCLOB` datatype
`boolean_val`	Modifies the default value for columns of the `BOOLEAN` datatype

Example D-1 Change the Current Value for Columns That Use the `NUMBER` Data Type

1. `EXEC DBMS_REDACT.UPDATE_FULL_REDACTION_VALUES (number_val => 7);`
2. Close the PDB:

   ```
   ALTER PLUGGABLE DATABASE <pdb_name> CLOSE IMMEDIATE;
   ```
3. Re-open the PDB:

   ```
   ALTER PLUGGABLE DATABASE <pdb_name> OPEN;
   ```

Usage Notes

See DBMS_REDACT Security Model for a list of the required privileges for this procedure.

D.5.10 UPDATE_POLICY_EXPRESSION Procedure

This procedure updates an Oracle Data Redaction named policy expression.

Syntax

```
DBMS_REDACT.UPDATE_POLICY_EXPRESSION (
   policy_expression_name          IN    VARCHAR2,
   expression                      IN    VARCHAR2,
   policy_expression_description   IN    VARCHAR2 := NULL);
```

Parameters

Table D-15 *UPDATE_POLICY_EXPRESSION Procedure Parameters*

Parameter	Description
`policy_expression_name`	Name of the policy expression
`expression`	Definition of the policy expression
`policy_expression_description`	Description of the policy expression

Exceptions

- `ORA-28082` - The parameter *parameter* is invalid.
- `ORA-28092` - The parameter *parameter* with value *value* has an error.

Usage Notes

See DBMS_REDACT Security Model for a list of the required privileges for this procedure.

You can find existing policy expressions by querying the `REDACTION_EXPRESSIONS` data dictionary view.

Example

```
BEGIN
   DBMS_REDACT.UPDATE_POLICY_EXPRESSION(
     policy_expression_name         => 'oe_redact_pol',
     expression                     => 'SYS_CONTEXT(''USERENV'',''SESSION_USER'') !=
''OE'''),
     policy_expression_description  => 'Updates policy expression for oe_redact_pol');
END;
```

E Managing Oracle Data Redaction Policies in Oracle Enterprise Manager

Oracle Enterprise Manager Cloud Control (Cloud Control) can manage Oracle Data Redaction policies and formats.

E.1 About Using Oracle Data Redaction in Oracle Enterprise Manager

Oracle Enterprise Manager Cloud Control provides an unified user interface for creating and managing Oracle Data Redaction policies.

You can create and manage sensitive column types directly from the Oracle Data Redaction pages. While you create a Data Redaction policy, Cloud Control uses sensitive column types to obtain the Oracle Data Redaction formats that are relevant to the column that you are redacting.

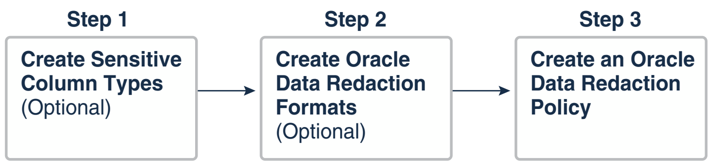
Note:

Ensure that you have the latest plug-in for Oracle Enterprise Manager.

E.2 Oracle Data Redaction Workflow

First, you should create sensitive column types and formats if necessary, and then create the Oracle Data Redaction policy afterward.

The following figure illustrates this process:


Step 1
Create Sensitive Column Types
(Optional)
Step 2
Create Oracle Data Redaction Formats
(Optional)
Step 3
Create an Oracle Data Redaction Policy

1. (Optional) If you want to map the database columns (that contain the data that you want to redact) to new sensitive column types, then create the required sensitive column types as described in Management of Sensitive Column Types in Enterprise Manager.
2. (Optional) If you want to redact the data (present in a particular database column) using a custom redaction format, then create the required redaction format as described in Creating a Custom Oracle Data Redaction Format Using Enterprise Manager.
3. Create an Oracle Data Redaction policy for the required database, as described in Creating an Oracle Data Redaction Policy Using Enterprise Manager.

Note:

When you create an Oracle Data Redaction policy, it is enabled by default. For information on how to disable an enabled redaction policy, see Enabling or Disabling an Oracle Data Redaction Policy in Enterprise Manager.

E.3 Management of Sensitive Column Types in Enterprise Manager

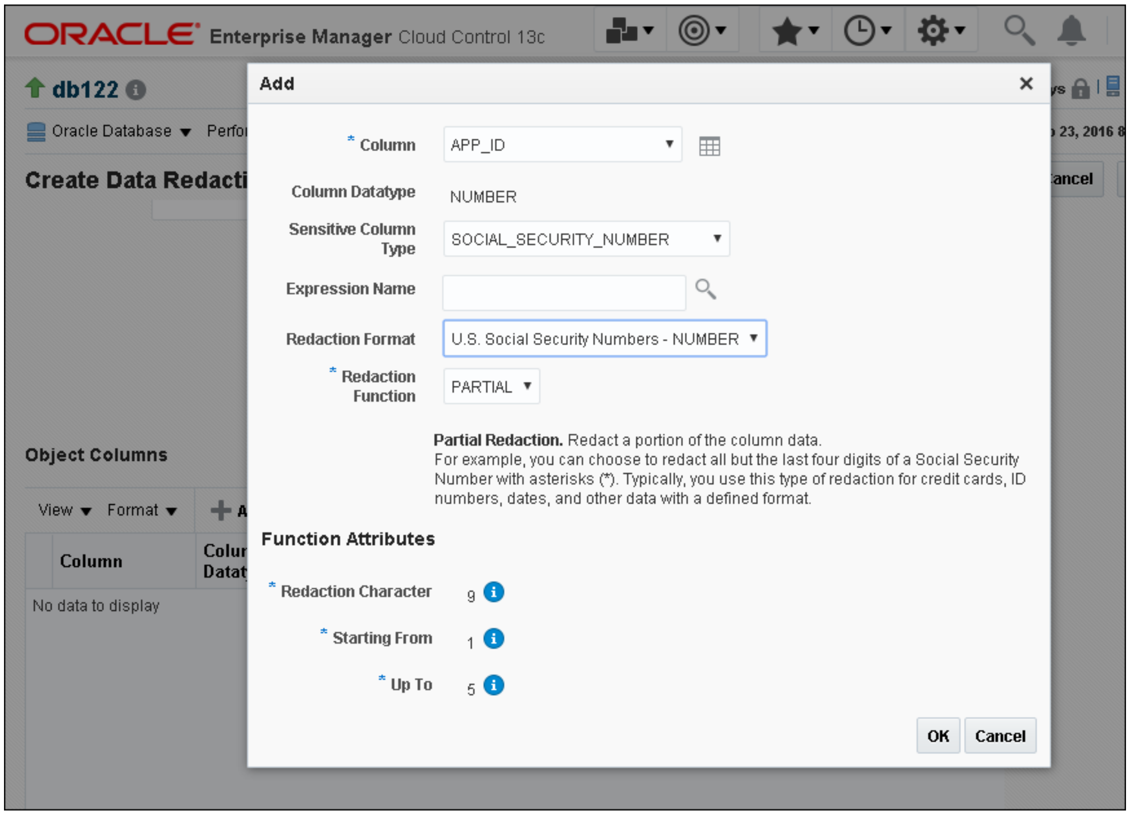
A sensitive column type categorizes table column sensitive information into a sensitive information type, such as credit card numbers.

Sensitive column types use a combination of the column name, column comments, and the data pattern defined using a regular expression to tag a column to a particular sensitive information type.

While you create Oracle Data Redaction policies, redaction formats are filtered on the basis of the chosen sensitive column type, thus saving time and effort. For example, if the database table column that you want to redact contains U.S. Social Security Numbers, and you select the `SOCIAL_SECURITY_NUMBER` sensitive column type for the column while adding it to the Oracle Data Redaction policy, the default redaction formats that you can use to redact the column data are filtered, and only the relevant redaction formats are displayed.

Figure E-1 illustrates the filtering of Oracle Data Redaction formats based on sensitive column types.

Figure E-1 Oracle Data Redaction Formats Filtered on the Basis of Sensitive Column Types



* Column: APP_ID
Column Datatype: NUMBER
Sensitive Column Type: SOCIAL_SECURITY_NUMBER
Expression Name:
Redaction Format: U.S. Social Security Numbers - NUMBER
* Redaction Function: PARTIAL

Partial Redaction. Redact a portion of the column data.
For example, you can choose to redact all but the last four digits of a Social Security Number with asterisks (*). Typically, you use this type of redaction for credit cards, ID numbers, dates, and other data with a defined format.

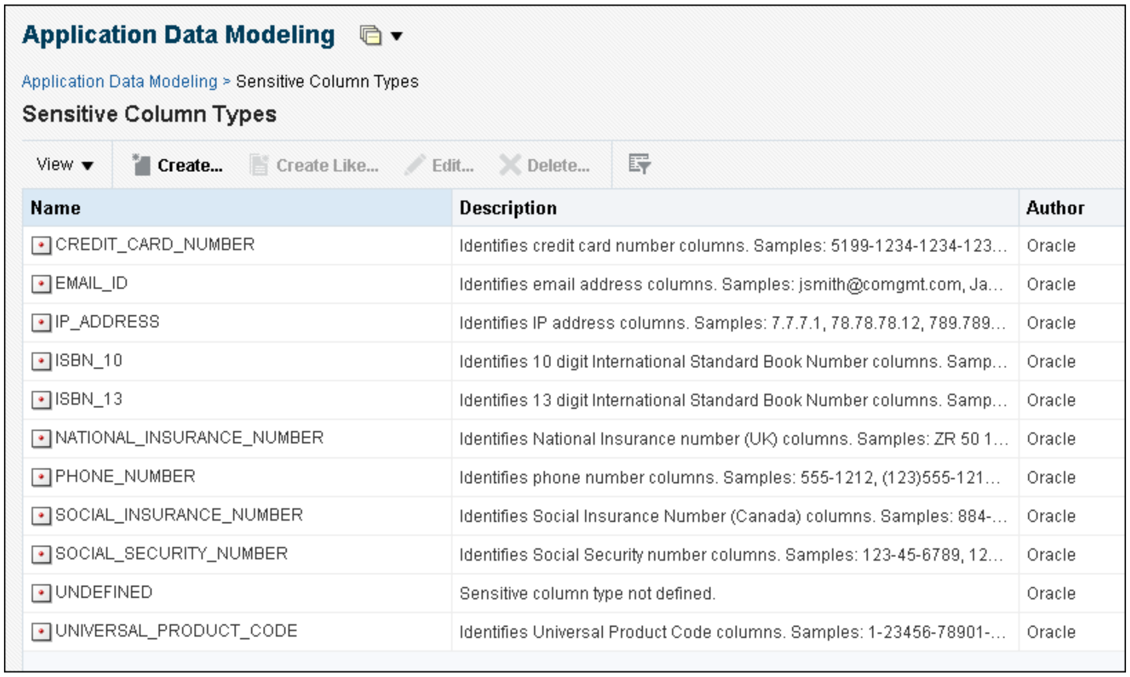
Function Attributes
* Redaction Character: 9
* Starting From: 1
* Up To: 5

OK Cancel

As part of the Application Data Modeling feature, Oracle provides a number of default sensitive column types that a database column can be mapped to.

Figure E-2 displays some of the default sensitive column types. To access this page, click **Manage Sensitive Column Types** on the Data Redaction Formats page.

Figure E-2 Default Sensitive Column Types



Application Data Modeling
Application Data Modeling > Sensitive Column Types
Sensitive Column Types
View
Create...
Create Like...
Edit...
Delete...

Name	Description	Author
CREDIT_CARD_NUMBER	Identifies credit card number columns. Samples: 5199-1234-1234-123...	Oracle
EMAIL_ID	Identifies email address columns. Samples: jsmith@comgmt.com, Ja...	Oracle
IP_ADDRESS	Identifies IP address columns. Samples: 7.7.7.1, 78.78.78.12, 789.789...	Oracle
ISBN_10	Identifies 10 digit International Standard Book Number columns. Samp...	Oracle
ISBN_13	Identifies 13 digit International Standard Book Number columns. Samp...	Oracle
NATIONAL_INSURANCE_NUMBER	Identifies National Insurance number (UK) columns. Samples: ZR 50 1...	Oracle
PHONE_NUMBER	Identifies phone number columns. Samples: 555-1212, (123)555-121...	Oracle
SOCIAL_INSURANCE_NUMBER	Identifies Social Insurance Number (Canada) columns. Samples: 884-...	Oracle
SOCIAL_SECURITY_NUMBER	Identifies Social Security number columns. Samples: 123-45-6789, 12...	Oracle
UNDEFINED	Sensitive column type not defined.	Oracle
UNIVERSAL_PRODUCT_CODE	Identifies Universal Product Code columns. Samples: 1-23456-78901-...	Oracle

If none of the default sensitive column types are suitable for the database column that contains the data that you want to redact, you can create a new sensitive column type, or create a sensitive column type that is based on an existing sensitive column type.

E.4 Managing Oracle Data Redaction Formats Using Enterprise Manager

Oracle Data Redaction provides redaction formats to be used directly within a redaction policy to redact data.

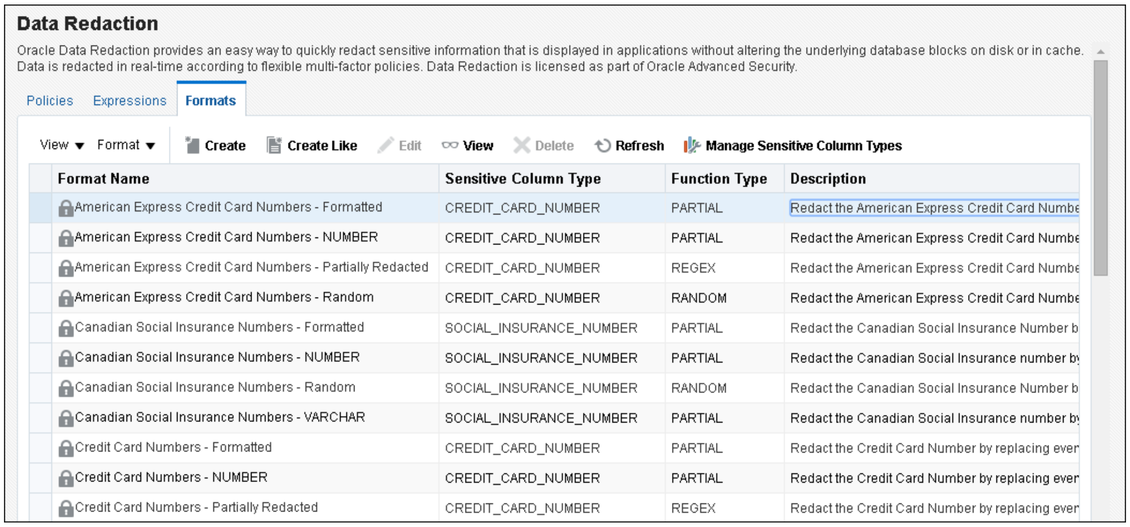
E.4.1 About Managing Oracle Data Redaction Formats Using Enterprise Manager

The Oracle Data Redaction formats are used for commonly redacted data, such as ID numbers, credit cards, or phone numbers.

You can use several default Oracle Data Redaction formats (previously known as Oracle Data Redaction templates). As an example of the Oracle Data Redaction formats, a set of Social Security Number formats enable you to quickly designate ways to redact Social Security Numbers, such as redacting the first five numbers of the Social Security Number.

Figure E-3 displays the default Oracle Data Redaction formats.

Figure E-3 Default Oracle Data Redaction Formats



Data Redaction

Oracle Data Redaction provides an easy way to quickly redact sensitive information that is displayed in applications without altering the underlying database blocks on disk or in cache. Data is redacted in real-time according to flexible multi-factor policies. Data Redaction is licensed as part of Oracle Advanced Security.

Format Name	Sensitive Column Type	Function Type	Description
American Express Credit Card Numbers - Formatted	CREDIT_CARD_NUMBER	PARTIAL	Redact the American Express Credit Card Numbe
American Express Credit Card Numbers - NUMBER	CREDIT_CARD_NUMBER	PARTIAL	Redact the American Express Credit Card Numbe
American Express Credit Card Numbers - Partially Redacted	CREDIT_CARD_NUMBER	REGEX	Redact the American Express Credit Card Numbe
American Express Credit Card Numbers - Random	CREDIT_CARD_NUMBER	RANDOM	Redact the American Express Credit Card Numbe
Canadian Social Insurance Numbers - Formatted	SOCIAL_INSURANCE_NUMBER	PARTIAL	Redact the Canadian Social Insurance Number b
Canadian Social Insurance Numbers - NUMBER	SOCIAL_INSURANCE_NUMBER	PARTIAL	Redact the Canadian Social Insurance number by
Canadian Social Insurance Numbers - Random	SOCIAL_INSURANCE_NUMBER	RANDOM	Redact the Canadian Social Insurance Number b
Canadian Social Insurance Numbers - VARCHAR	SOCIAL_INSURANCE_NUMBER	PARTIAL	Redact the Canadian Social Insurance number by
Credit Card Numbers - Formatted	CREDIT_CARD_NUMBER	PARTIAL	Redact the Credit Card Number by replacing ever
Credit Card Numbers - NUMBER	CREDIT_CARD_NUMBER	PARTIAL	Redact the Credit Card Number by replacing ever
Credit Card Numbers - Partially Redacted	CREDIT_CARD_NUMBER	REGEX	Redact the Credit Card Number by replacing ever

Each default Oracle Data Redaction format consists of a specific redaction function that determines the redacted output when the redaction format is used in an Oracle Data Redaction policy. For example, the `Credit Card Numbers - NUMBER` default redaction format replaces the first twelve digits of the column data with the digit 0, when it is used in an Oracle Data Redaction policy. That is, if the column data is `5555555555554444`, the redacted output will be `0000000000004444`.

E.4.2 Creating a Custom Oracle Data Redaction Format Using Enterprise Manager

You can create and save custom Oracle Data Redaction formats using Enterprise Manager Cloud Control.

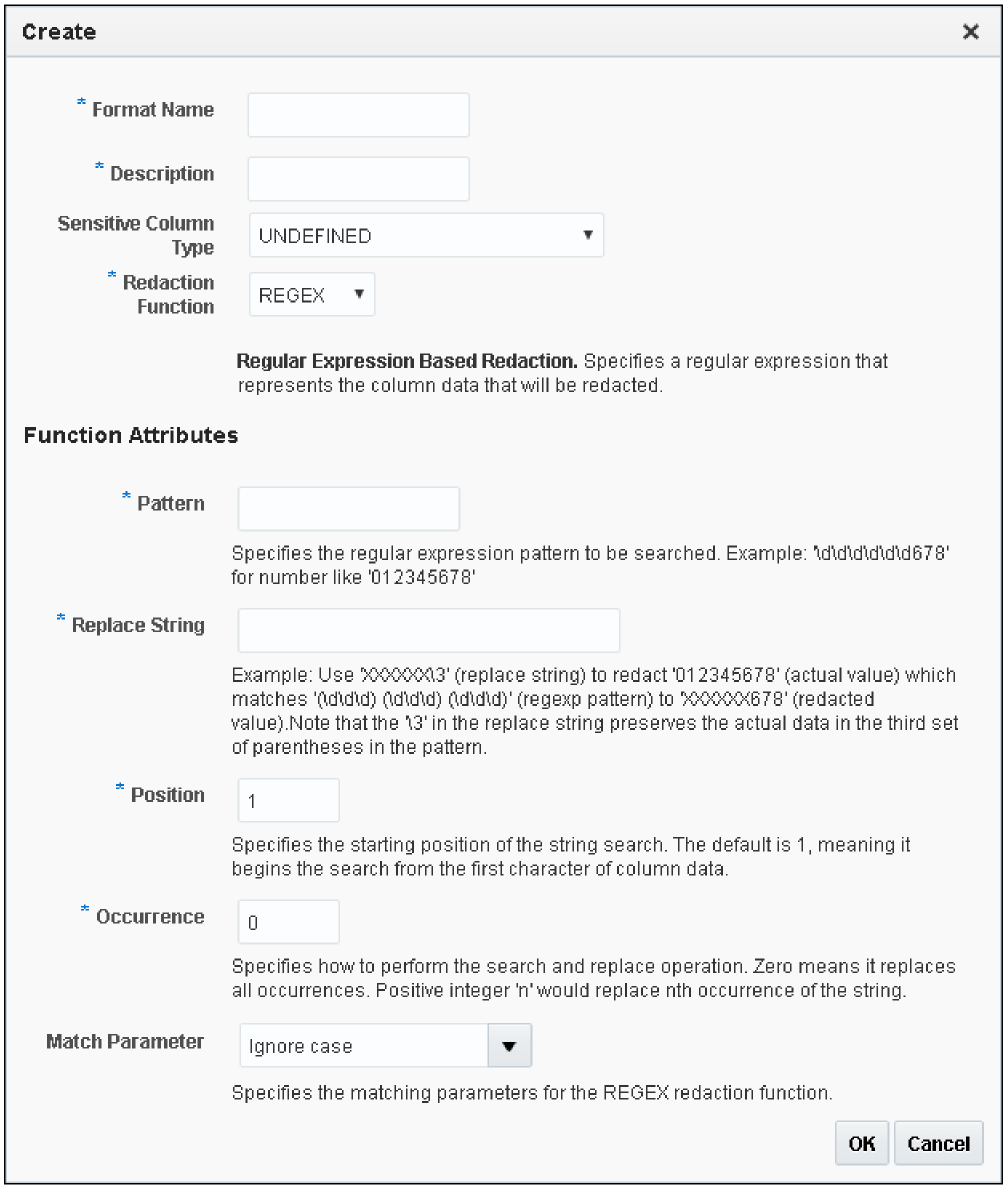
1. Log into Oracle Enterprise Manager Cloud Control as either user `SYSTEM` or `SYSMAN`.

   The URL is as follows:

   ```
   https://host:port/em
   ```

2. From the **Targets** menu, select **Databases**.
3. Select **Search List**, then click the name of a database target.
4. On the home page of the database target, from the **Security** menu, select **Data Redaction**.
5. If you are prompted to do so, then log in to the database as a user who has the `EXECUTE` privilege on the `DBMS_REDACT` PL/SQL package and the `ADMINISTER REDACTION POLICY` system or schema privilege.
6. Select the **Formats** tab.
7. Do one of the following:
   - To create a new redaction format, click **Create**.
   - To create a redaction format that is based on a default format, select the format and then click **Create Like**.

   If you select **Create**, then the following dialog box appears:



* Format Name

* Description

Sensitive Column Type: UNDEFINED

* Redaction Function: REGEX

Regular Expression Based Redaction. Specifies a regular expression that represents the column data that will be redacted.

Function Attributes

* Pattern

Specifies the regular expression pattern to be searched. Example: '\d\d\d\d\d\d678' for number like '012345678'

* Replace String

Example: Use 'XXXXXX\3' (replace string) to redact '012345678' (actual value) which matches '(\d\d\d) (\d\d\d) (\d\d\d)' (regexp pattern) to 'XXXXXX678' (redacted value).Note that the '\3' in the replace string preserves the actual data in the third set of parentheses in the pattern.

* Position: 1

Specifies the starting position of the string search. The default is 1, meaning it begins the search from the first character of column data.

* Occurrence: 0

Specifies how to perform the search and replace operation. Zero means it replaces all occurrences. Positive integer 'n' would replace nth occurrence of the string.

Match Parameter: Ignore case

Specifies the matching parameters for the REGEX redaction function.

8. Provide a name and a description for the redaction format that you want to create.

   If you want to map the redaction format to a particular sensitive column type (such that the created redaction format can be used to redact the data of a column that is associated with the sensitive column type), then select a value for **Sensitive Column Type**.

   Select the function that the format should use to redact the column data. For **Redaction Function**, select as follows:

   - **FULL** if the format should redact the entire column data.
   - **PARTIAL** if the format should redact only a part of the column data. Ensure that you provide the function attributes, as well as the data type that you want to use the redaction format for.
   - **REGEX** if the format should redact data based on a regular expression. Ensure that you provide the function attributes.
   - **RANDOM** if the format should redact data in a random manner, using randomly generated values

- **NONE** if the format will be used to only test the definition of a redaction policy, and not redact any column data

> **Note:**
>
> The Nullify redaction type is not supported in Oracle Enterprise Manager.

9. Click **OK** to create and save the custom redaction format.

   This format now can be used to create a redaction policy.

Related Topics

- Oracle Data Redaction Features and Capabilities
- Creating an Oracle Data Redaction Policy Using Enterprise Manager
  You can create an Oracle Data Redaction policy using Enterprise Manager Cloud Control.

E.4.3 Editing a Custom Oracle Data Redaction Format Using Enterprise Manager

You can edit custom Oracle Data Redaction formats using Enterprise Manager Cloud Control, but not in SQL*Plus.

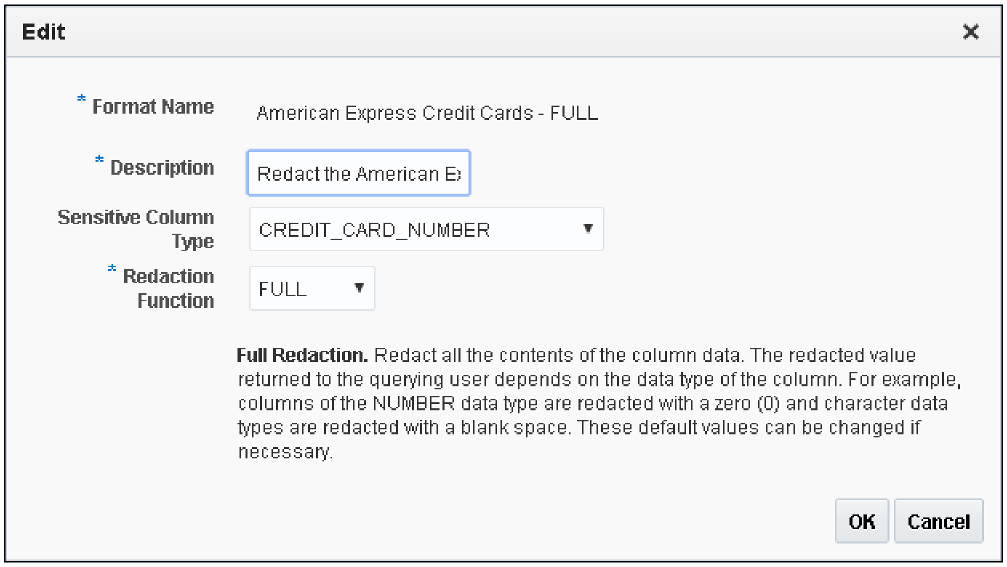
1. Log into Oracle Enterprise Manager Cloud Control as either user `SYSTEM` or `SYSMAN`.

   The URL is as follows:

   https://*host*:*port*/em

2. From the **Targets** menu, select **Databases**.
3. Select **Search List**, then click the name of a database target.
4. On the home page of the database target, from the **Security** menu, select **Data Redaction**.
5. If you are prompted to do so, then log in to the database as a user who has the `EXECUTE` privilege on the `DBMS_REDACT` PL/SQL package and the `ADMINISTER REDACTION POLICY` system or schema privilege.
6. Select the **Formats** tab.
7. Select the custom redaction format that you want to edit, then click **Edit.**

   A dialog box similar to the following appears:



* Format Name: American Express Credit Cards - FULL

* Description: Redact the American E›

Sensitive Column Type: CREDIT_CARD_NUMBER

* Redaction Function: FULL

Full Redaction. Redact all the contents of the column data. The redacted value returned to the querying user depends on the data type of the column. For example, columns of the NUMBER data type are redacted with a zero (0) and character data types are redacted with a blank space. These default values can be changed if necessary.

8. (Optional) Choose to edit the format description, sensitive column type, redaction function, and the redaction function attributes.
9. Click **OK** to save the edited format.

E.4.4 Viewing Oracle Data Redaction Formats Using Enterprise Manager

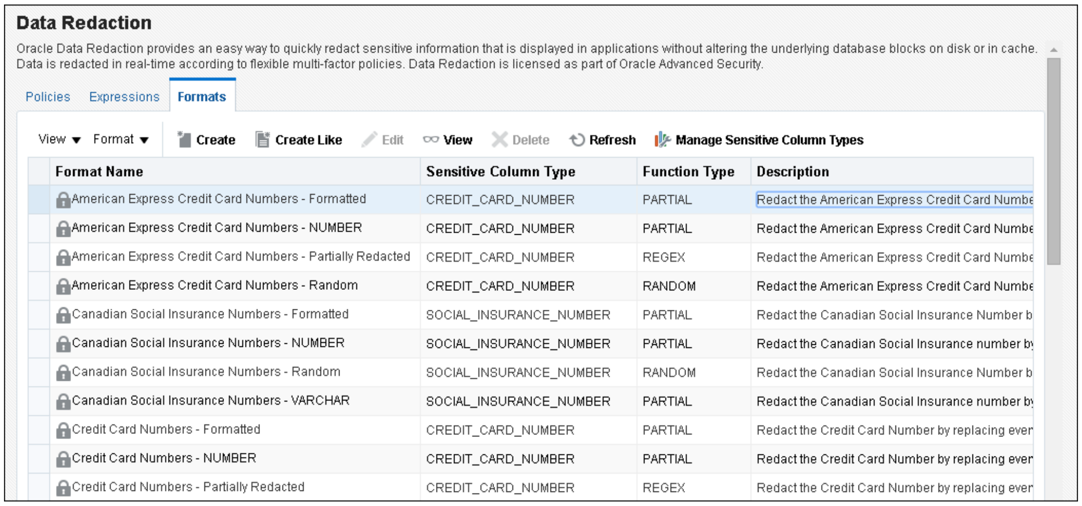
Enterprise Manager Cloud Control displays the details of the Oracle-supplied and custom Oracle Data Redaction formats.

1. Log into Oracle Enterprise Manager Cloud Control as either user `SYSTEM` or `SYSMAN`.

   The URL is as follows:

   https://*host*:*port*/em
2. **Search List**
3. From the **Targets** menu, select **Databases**.
4. Select , then click the name of a database target.
5. On the home page of the database target, from the **Security** menu, select **Data Redaction**.
6. Log in to the database as a user who has privileges to view information in Oracle Data Redaction data dictionary views.
7. Select the **Formats** tab.

   The Data Redaction Formats page appears, similar to the following page.



Data Redaction

Oracle Data Redaction provides an easy way to quickly redact sensitive information that is displayed in applications without altering the underlying database blocks on disk or in cache. Data is redacted in real-time according to flexible multi-factor policies. Data Redaction is licensed as part of Oracle Advanced Security.

Format Name	Sensitive Column Type	Function Type	Description
American Express Credit Card Numbers - Formatted	CREDIT_CARD_NUMBER	PARTIAL	Redact the American Express Credit Card Numbe
American Express Credit Card Numbers - NUMBER	CREDIT_CARD_NUMBER	PARTIAL	Redact the American Express Credit Card Numbe
American Express Credit Card Numbers - Partially Redacted	CREDIT_CARD_NUMBER	REGEX	Redact the American Express Credit Card Numbe
American Express Credit Card Numbers - Random	CREDIT_CARD_NUMBER	RANDOM	Redact the American Express Credit Card Numbe
Canadian Social Insurance Numbers - Formatted	SOCIAL_INSURANCE_NUMBER	PARTIAL	Redact the Canadian Social Insurance Number b
Canadian Social Insurance Numbers - NUMBER	SOCIAL_INSURANCE_NUMBER	PARTIAL	Redact the Canadian Social Insurance number by
Canadian Social Insurance Numbers - Random	SOCIAL_INSURANCE_NUMBER	RANDOM	Redact the Canadian Social Insurance Number b
Canadian Social Insurance Numbers - VARCHAR	SOCIAL_INSURANCE_NUMBER	PARTIAL	Redact the Canadian Social Insurance number by
Credit Card Numbers - Formatted	CREDIT_CARD_NUMBER	PARTIAL	Redact the Credit Card Number by replacing ever
Credit Card Numbers - NUMBER	CREDIT_CARD_NUMBER	PARTIAL	Redact the Credit Card Number by replacing ever
Credit Card Numbers - Partially Redacted	CREDIT_CARD_NUMBER	REGEX	Redact the Credit Card Number by replacing ever

8. Select the required redaction format, then click **View**.

E.4.5 Deleting a Custom Oracle Data Redaction Format Using Enterprise Manager

You can delete custom Oracle Data Redaction formats using Enterprise Manager Cloud Control.

You can only delete custom Oracle Data Redaction formats, and not the redaction formats that are provided by Oracle.

1. Log into Oracle Enterprise Manager Cloud Control as either user `SYSTEM` or `SYSMAN`.

   The URL is as follows:

   https://*host*:*port*/em

2. From the **Targets** menu, select **Databases**.
3. Select **Search List**, then click the name of a database target.
4. On the home page of the database target, from the **Security** menu, select **Data Redaction**.
5. If you are prompted to do so, then log in to the database as a user who has the `EXECUTE` privilege on the `DBMS_REDACT` PL/SQL package and the `ADMINISTER REDACTION POLICY` system or schema privilege.
6. Select the **Formats** tab.
7. Select the custom redaction format that you want to delete, and then click **Delete**.
8. In the Confirmation dialog box, click **Yes** or **No**.

E.5 Managing Oracle Data Redaction Named Policy Expressions Using Enterprise Manager

You can create, edit, view, and delete Oracle Data Redaction named policy expressions in Enterprise Manager Cloud Control.

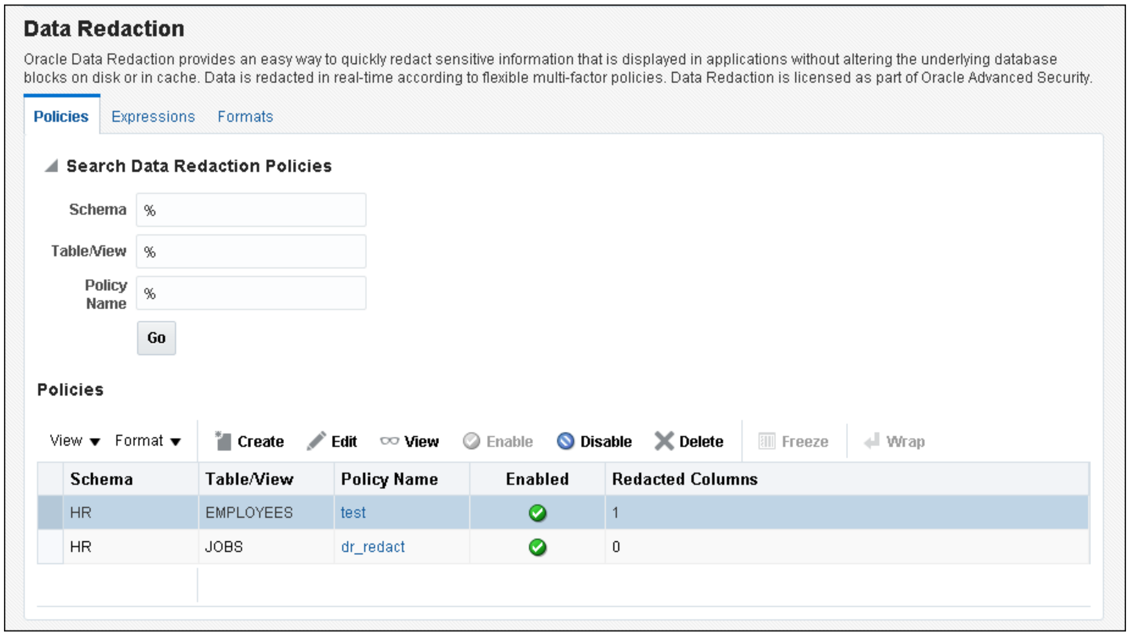
E.5.1 About Managing Oracle Data Redaction Policies Using Enterprise Manager

Use the Data Redaction page in Cloud Control to manage Oracle Data Redaction policies.

To redact the data present in a particular database table or view column, you must create an Oracle Data Redaction policy. Data is redacted using a redaction format that is specified by the Oracle Data Redaction policy. To redact data, you can use any of the Oracle-supplied redaction formats, or create and use a custom redaction format. If the table or view column that contains the data that you want to redact is mapped to a sensitive column type, Oracle uses the mapping to recommend suitable redaction formats for the data. Thus, Oracle Data Redaction policies encapsulate database schemas, database table and view columns, sensitive column types, and Oracle Data Redaction formats.

Figure E-4 shows the Data Redaction page, which enables you to create and manage Oracle Data Redaction policies in Cloud Control.

Figure E-4 Oracle Data Redaction Policies Page


Data Redaction

Oracle Data Redaction provides an easy way to quickly redact sensitive information that is displayed in applications without altering the underlying database blocks on disk or in cache. Data is redacted in real-time according to flexible multi-factor policies. Data Redaction is licensed as part of Oracle Advanced Security.

Policies | Expressions | Formats

Search Data Redaction Policies

Schema	%
Table/View	%
Policy Name	%

Go

Policies

Schema	Table/View	Policy Name	Enabled	Redacted Columns
HR	EMPLOYEES	test	✓	1
HR	JOBS	dr_redact	✓	0

E.5.2 Creating an Oracle Data Redaction Policy Using Enterprise Manager

You can create an Oracle Data Redaction policy using Enterprise Manager Cloud Control.

1. Log into Oracle Enterprise Manager Cloud Control as either user `SYSTEM` or `SYSMAN`.

   The URL is as follows:

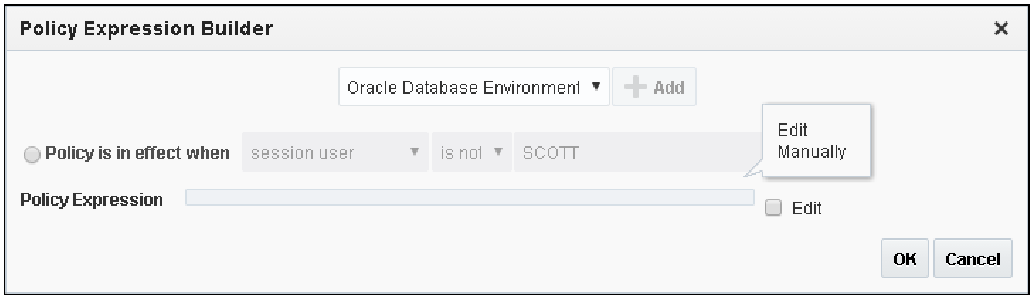
   ```
   https://host:port/em
   ```

2. From the **Targets** menu, select **Databases**.
3. Select **Search List**, then click the name of a database target for which you want to create an Oracle Data Redaction policy.

4. On the home page of the database target, from the **Security** menu, select **Data Redaction**.
5. If you are prompted to do so, then log in to the database as a user who has the `EXECUTE` privilege on the `DBMS_REDACT` PL/SQL package and the `ADMINISTER REDACTION POLICY` system or schema privilege.
6. In the Policies section of the Policies tab, select **Create**.

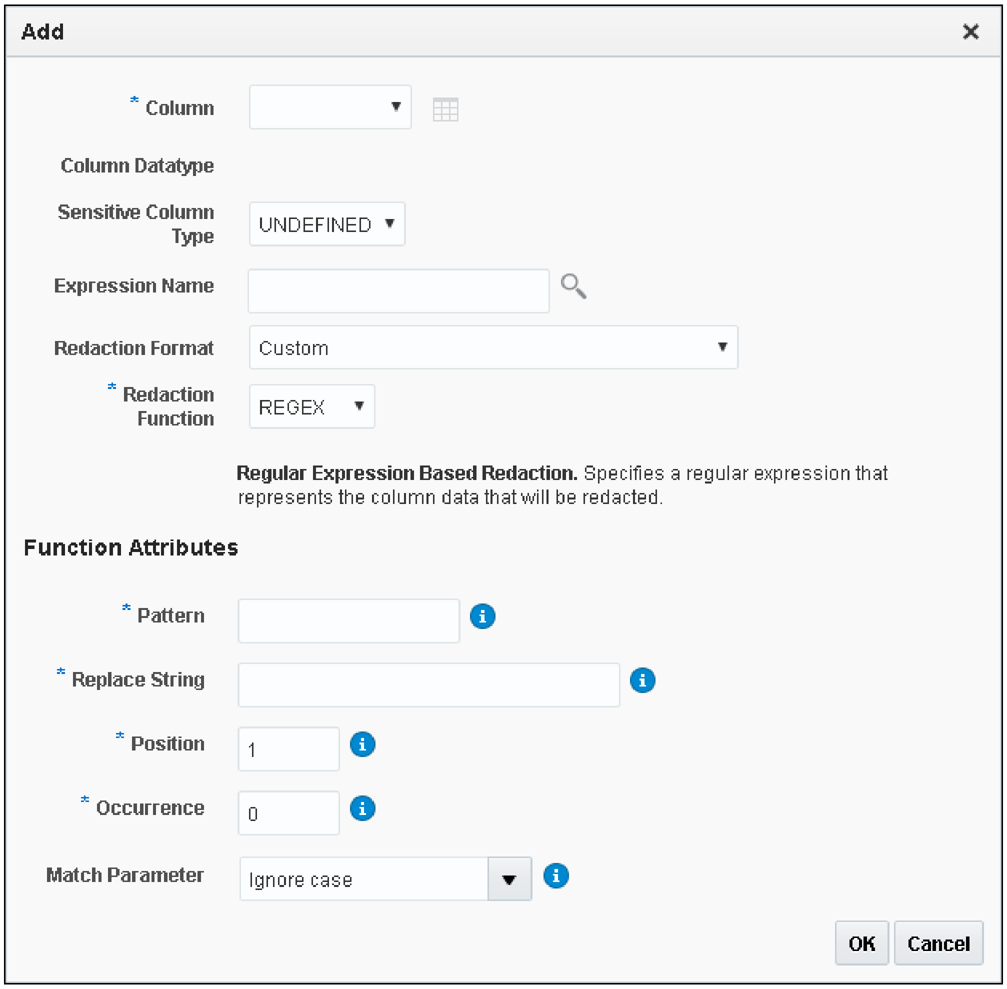
   If this is the first time that you are creating a Data Redaction policy, then the Data Redaction setup for enabling column sensitive type discovery dialog box appears. This feature enables the use of column sensitive type discovery for Data Redaction policies.

   To accomplish this, Enterprise Manager creates the `GET_COL_DATA_SENSITIVE_TYPES` procedure in the `DBSNMP` schema. To perform a sensitive type discovery for a selected column while creating an Oracle Data Redaction policy, a user must have the `EXECUTE` privilege on the `DBSNMP.GET_COL_DATA_SENSITIVE_TYPES` procedure. If the database is protected by Oracle Database Vault, then ensure that any users who must create Data Redaction policies are participants to realms that protect the `DBSNMP` schema.
7. If the Data Redaction setup for enabling column sensitive type discovery dialog box appears and if the current login user does not have the correct requirements, select a credential of a user who has the `EXECUTE` privilege on `DBSNMP.GET_COL_DATA_SENSITIVE_TYPES`. Then click `OK`.
8. On the Create Data Redaction Policy page, enter the following information:
   - **Schema**: Enter (or search for) the name of the schema that contains the data you want to redact.
   - **Table/View**: Enter (or search for) the table or field that contains the column you want to redact.
   - **Policy Name**: Enter a name for the policy, such as `emp_wages_pol`.
   - **Default Expression**: Enter the default expression. The default setting is 1=1, which means that the policy will always be enforced. If you are not familiar with the components of a policy expression, then click the pencil icon beside the **Policy Expression** field to use Policy Expression Builder. Select **Policy is in effect when**, select the required conditions, then click **Add**. Click **Edit** if you want to edit the policy expression manually. After building the required policy expression, click **OK**. The Policy Expression Builder appears as follows:


Policy Expression Builder
Oracle Database Environment ▼ Add
Policy is in effect when session user ▼ is not ▼ SCOTT
Edit Manually
Policy Expression
Edit
OK Cancel

9. In the Object Columns section, click **Add** to add a table or view column to the redaction policy.

   A dialog box similar to the following appears:



Regular Expression Based Redaction. Specifies a regular expression that represents the column data that will be redacted.

The redaction policy is applied only on the table or view columns that are added to it.

10. From the **Column** menu, select the table or view column to which you want to apply the redaction policy.

    To the right of the **Column** menu is an icon that you can click to view the contents of the selected column.

    If the column contains sensitive data and has been mapped to a sensitive column type, then from the **Sensitive Column Type** menu, select the sensitive column type that it has been mapped to. If the search pattern in the **Sensitive Column Type** menu matches, then the sensitive column type is selected by default. For example, for a column listing credit card numbers, if there is a match, then the menu will list **Undefined** and **CREDIT_CARD_TYPE**. If there is no sensitive column type created, then the default **Sensitive Column Type** menu listing is only **Undefined**.

11. From the **Redaction Format** menu, select the redaction format that you want to use.

    The drop-down list is populated with the Oracle Database-supplied redaction formats, as well as the custom redaction formats that you have created and saved.
    If you do not want to use a pre-defined redaction format (that is, an Oracle-Database supplied redaction format, or a custom redaction format that you have created), and instead want to specify the redaction details while creating the redaction policy, select **CUSTOM** for **Redaction Format**.

    The Add dialog box adjusts to accommodate the type of redaction format and function that you select. For example, if you select the **CUSTOM** redaction format and the **REGEX** redaction function, then the Function Attributes region appears in the dialog box.

12. From the **Redaction Function** menu, select the function that you want to use to redact the column data.

    Select **FULL** if you want to redact the entire column data, **PARTIAL** if you want to redact only a part of the column data, **REGEX** if you want to redact the column data based on a regular expression, **RANDOM** if you want to redact the column data in a random manner, using randomly generated values, or **NONE** if you only want to test the definition of the redaction policy, and not redact any column data. Note that all the redaction functions may not be applicable for a particular redaction format. The drop-down list displays only the redaction functions that are applicable for the selected redaction format. (Note that the Nullify redaction type is not supported in Oracle Enterprise Manager.)

    If you selected **CUSTOM** for **Redaction Format** in the previous step, and **PARTIAL** or **REGEX** for **Redaction Function,** ensure that you specify the function attributes.

13. Click **OK**.
14. Repeat these steps starting with Step 8 for all the columns that you want to add to the redaction policy.
15. On the Create Data Redaction Policy page, click **OK** to create the data redaction policy.

    When you create an Oracle Data Redaction policy, it is enabled by default.

Related Topics

- Creating a Custom Oracle Data Redaction Format Using Enterprise Manager
  You can create and save custom Oracle Data Redaction formats using Enterprise Manager Cloud Control.
- Oracle Data Redaction Features and Capabilities
- Enabling or Disabling an Oracle Data Redaction Policy in Enterprise Manager
  An Oracle Data Redaction policy is executed at run time only if it is enabled. When you create an Oracle Data Redaction policy, it is enabled by default.

E.5.3 Editing an Oracle Data Redaction Policy Using Enterprise Manager

You can edit an Oracle Data Redaction policy using Enterprise Manager Cloud Control.

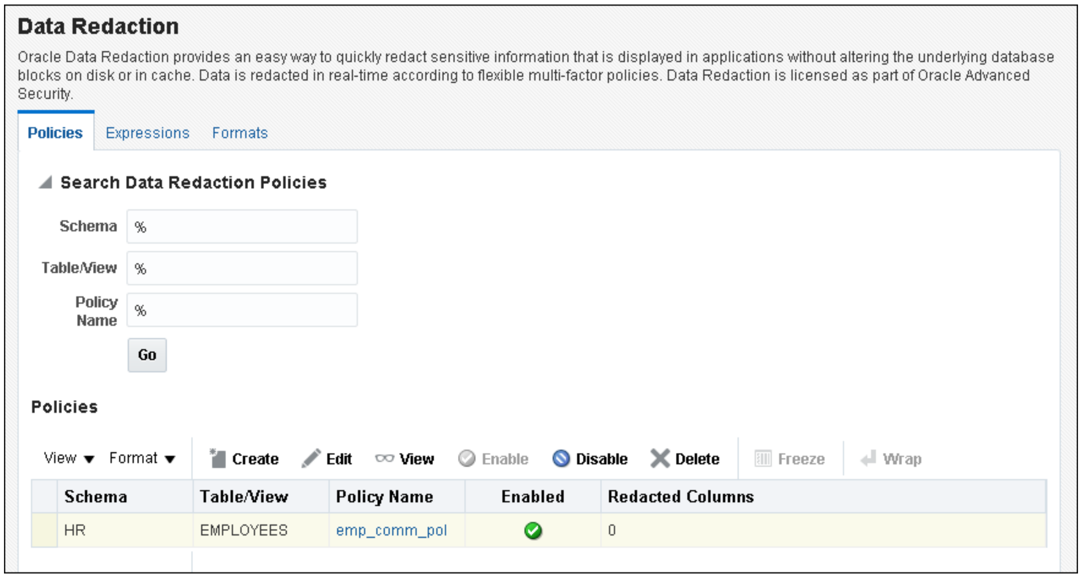
1. Log into Oracle Enterprise Manager Cloud Control as either user `SYSTEM` or `SYSMAN`.

   The URL is as follows:

   ```
   https://host:port/em
   ```

2. From the **Targets** menu, select **Databases**.
3. Select **Search List**, then search for and click the name of the database target for which the Oracle Data Redaction policy that you want to edit was created.
4. On the home page of the database target, from the **Security** menu, select **Data Redaction**.
5. If you are prompted to do so, then log in to the database as a user who has the `EXECUTE` privilege on the `DBMS_REDACT` PL/SQL package and the `ADMINISTER REDACTION POLICY` system or schema privilege.

6. In the Policies section of the **Policies** tab, select the redaction policy that you want to edit, then click **Edit**.



Data Redaction

Oracle Data Redaction provides an easy way to quickly redact sensitive information that is displayed in applications without altering the underlying database blocks on disk or in cache. Data is redacted in real-time according to flexible multi-factor policies. Data Redaction is licensed as part of Oracle Advanced Security.

Schema	%
Table/View	%
Policy Name	%

Schema	Table/View	Policy Name	Enabled	Redacted Columns
HR	EMPLOYEES	emp_comm_pol	✓	0

7. On the Edit Data Redaction Policy page, choose to edit the policy expression, add new columns to the redaction policy, modify the redaction details of a column that is part of the policy, or delete a column from the redaction policy.

   You can do the following:

   - To add a new column to the redaction policy, in the Object Columns section, click **Add,** select the table or view column that you want to add, then specify the redaction details.
   - To modify the redaction details of a column that is part of the policy, select the column, click **Modify,** then edit the redaction details.
   - To delete a column from the redaction policy, select the column, then click **Delete**.

8. On the Edit Data Redaction Policy page, after editing the required fields, click **OK** to save and enable the edited redaction policy.

E.5.4 Viewing Oracle Data Redaction Policy Details Using Enterprise Manager

You can find Oracle Data Redaction policy details such as whether the policy is enabled by using Enterprise Manager Cloud Control.

1. Log into Oracle Enterprise Manager Cloud Control as either user `SYSTEM` or `SYSMAN`.

   The URL is as follows:

   https://*host*:*port*/em

2. From the **Targets** menu, select **Databases**.
3. Select **Search List**, then search for and click the name of the database target for which the Oracle Data Redaction policy that you want to view was created.
4. On the home page of the database target, from the **Security** menu, select **Data Redaction**.
5. Log in to the database as a user who has privileges to view information in Oracle Data Redaction data dictionary views.

6. In the Policies section of the Policies tab, do one of the following:
   - Select the name of the policy in the table.
   - Select the required redaction policy, then click **View**.
7. To exit, click **OK**.

E.5.5 Enabling or Disabling an Oracle Data Redaction Policy in Enterprise Manager

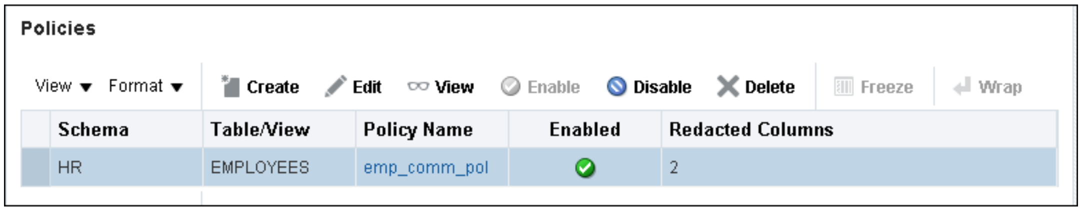
An Oracle Data Redaction policy is executed at run time only if it is enabled. When you create an Oracle Data Redaction policy, it is enabled by default.

You can disable an enabled redaction policy, or enable a disabled redaction policy using Enterprise Manager Cloud Control.

1. Log into Oracle Enterprise Manager Cloud Control as either user `SYSTEM` or `SYSMAN`.

   The URL is as follows:

   `https://host:port/em`
2. From the **Targets** menu, select **Databases**.
3. Select **Search List**, then search for and click the name of the database target for which the Oracle Data Redaction policy that you want to enable or disable was created.
4. On the home page of the database target, from the **Security** menu, select **Data Redaction**.
5. If you are prompted to do so, then log in to the database as a user who has the `EXECUTE` privilege on the `DBMS_REDACT` PL/SQL package and the `ADMINISTER REDACTION POLICY` system or schema privilege.
6. In the Policies section of the Policies tab, select the redaction policy that you want to enable or disable, and then click **Enable** or **Disable**.



Schema	Table/View	Policy Name	Enabled	Redacted Columns
HR	EMPLOYEES	emp_comm_pol	✓	2

7. In the Confirmation dialog box, click **Yes** or **No**.

E.5.6 Deleting an Oracle Data Redaction Policy Using Enterprise Manager

You can delete an Oracle Data Redaction policy using Enterprise Manager Cloud Control.

1. Log into Oracle Enterprise Manager Cloud Control as either user `SYSTEM` or `SYSMAN`.

   The URL is as follows:

   `https://host:port/em`
2. From the **Targets** menu, select **Databases**.
3. Select **Search List**, then search for and click the name of the database target for which the Oracle Data Redaction policy that you want to delete was created.

4. On the home page of the database target, from the **Security** menu, select **Data Redaction**.
5. Log in to the database, if you are prompted to do so.

   Ensure that you log in to the database as a user that has the `EXECUTE` privilege on the `DBMS_REDACT` PL/SQL package and the `ADMINISTER REDACTION POLICY` system or schema privilege.
6. In the Policies section of the Policies tab, select the redaction policy that you want to delete, and then click **Delete.**
7. In the Confirmation dialog box, click **Yes** or **No**.

E.6 Managing Data Redaction Named Policy Expressions Using Enterprise Manager

You can manage Oracle Data Redaction named policy expressions in Enterprise Manager Cloud Control.

E.6.1 About Data Redaction Named Policy Expressions in Enterprise Manager

You can create and apply an Oracle Data Redaction named policy expression to multiple columns in tables and views in Oracle Enterprise Manager Cloud Control.

When you modify the policy expression, the change is reflected in all redacted columns in the database instance that use the policy expression. Cloud Control enables you to create, edit, view, apply to columns, and delete policy expressions.

Related Topics

- Creating and Managing Multiple Named Policy Expressions
  A named, centrally managed Oracle Data Redaction policy expression can be used in multiple redaction policies and applied to multiple tables or views.

E.6.2 Creating a Data Redaction Named Policy Expression in Enterprise Manager

You can create and apply an Oracle Data Redaction named policy expression using Enterprise Manager Cloud Control.

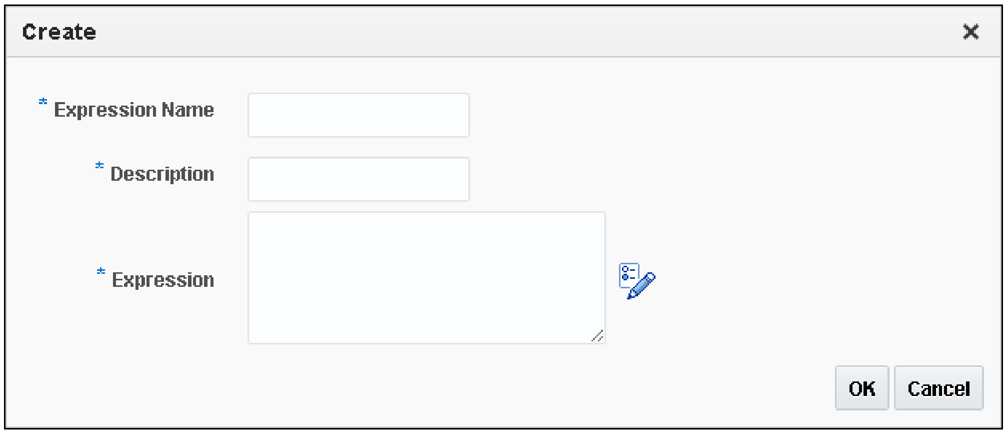
1. Log into Oracle Enterprise Manager Cloud Control as either user `SYSTEM` or `SYSMAN`.

   The URL is as follows:

   https://*host*:*port*/em
2. From the **Targets** menu, select **Databases**.
3. Select **Search List**, then click the name of a database target for which you want to create an Oracle Data Redaction policy.
4. On the home page of the database target, from the **Security** menu, select **Data Redaction**.

5. If you are prompted to do so, then log in to the database as a user who has the `EXECUTE` privilege on the `DBMS_REDACT` PL/SQL package and the `ADMINISTER REDACTION POLICY` system or schema privilege.
6. On the Oracle Data Redaction page, select the **Expressions** tab.
7. Click **Create**.

   The Create dialog box appears.



8. In the Create dialog box, enter the following information:
   - **Expression Name**: Enter a name for the policy expression. Existing policy expressions are listed on the Data Redaction page.
   - **Description**: Enter a brief description of the policy.
   - **Expression**: Enter the expression. For more complex expressions, such as applying or exempting the policy from specific users, click the Policy Expression Builder icon at the right of the **Expression** field. Click **OK** in the Policy Expression Builder to create the expression.
9. Click **OK** in the Create dialog box.

   After you create the policy expression, it is listed in the Data Redaction page and ready to be associated with a Data Redaction policy.
10. In the Data Redaction page, select the **Policies** tab.
11. Under Policies, select the row for the policy that redacts the column to which you want to apply the policy expression, and then click **Edit**.
12. Under Object Columns, select the column that you want and then click the **Modify** button.
13. In the Modify dialog box, select the expression from the **Expression Name** list.
14. Click **OK**, and then click **OK** again in the Edit Data Redaction Policy dialog box.

E.6.3 Editing a Data Redaction Named Policy Expression in Enterprise Manager

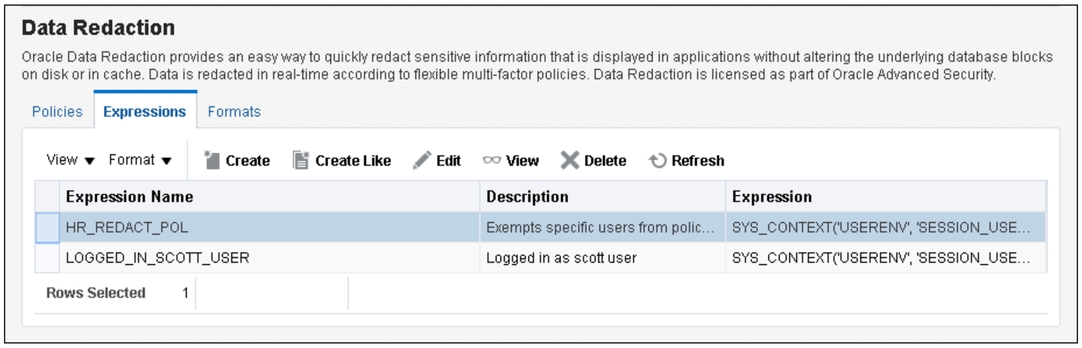
You can edit an Oracle Data Redaction named policy expression using Enterprise Manager Cloud Control.

1. Log into Oracle Enterprise Manager Cloud Control as either user `SYSTEM` or `SYSMAN`.

   The URL is as follows:

   ```
   https://host:port/em
   ```

2. From the **Targets** menu, select **Databases**.
3. Select **Search List**, then click the name of a database target for which you want to create an Oracle Data Redaction policy.
4. On the home page of the database target, from the **Security** menu, select **Data Redaction**.
5. If you are prompted to do so, then log in to the database as a user who has the `EXECUTE` privilege on the `DBMS_REDACT` PL/SQL package and the `ADMINISTER REDACTION POLICY` system or schema privilege.
6. On the Oracle Data Redaction page, select the **Expressions** tab.
7. Select the policy expression that you want to edit and then click **Edit**.



Oracle Data Redaction provides an easy way to quickly redact sensitive information that is displayed in applications without altering the underlying database blocks on disk or in cache. Data is redacted in real-time according to flexible multi-factor policies. Data Redaction is licensed as part of Oracle Advanced Security.

Expression Name	Description	Expression
HR_REDACT_POL	Exempts specific users from polic...	SYS_CONTEXT('USERENV', 'SESSION_USE...
LOGGED_IN_SCOTT_USER	Logged in as scott user	SYS_CONTEXT('USERENV', 'SESSION_USE...

Rows Selected 1

8. In the Edit dialog box, modify the **Description** and **Expression** fields as necessary. For more complex expressions, click the Policy Expression Builder icon, and then click **OK** after you have recreated the expression.
9. Click **OK** in the Edit dialog box.

E.6.4 Viewing Data Redaction Named Policy Expressions in Enterprise Manager

You can view Oracle Data Redaction named policy expressions using Enterprise Manager Cloud Control.

1. Log into Oracle Enterprise Manager Cloud Control as either user `SYSTEM` or `SYSMAN`.

   The URL is as follows:

   https://*host*:*port*/em

2. From the **Targets** menu, select **Databases**.
3. Select **Search List**, then click the name of a database target for which you want to create an Oracle Data Redaction policy.
4. On the home page of the database target, from the **Security** menu, select **Data Redaction**.
5. Log in to the database as a user who has privileges to view information in Oracle Data Redaction data dictionary views.
6. On the Oracle Data Redaction page, select the **Expressions** tab.
7. Select the policy expression that you want to view and then click **View**.

   The View dialog box appears, showing the definition of the policy expression.

8. Click **OK** to exit the View dialog box.

E.6.5 Deleting a Data Redaction Named Policy Expression in Enterprise Manager

You can delete Oracle Data Redaction named policy expressions using Enterprise Manager Cloud Control.

The deletion process first dissociates the policy expression from all columns to which it is applied.

1. Log into Oracle Enterprise Manager Cloud Control as either user `SYSTEM` or `SYSMAN`.

   The URL is as follows:

   `https://`*`host`*`:`*`port`*`/em`

2. From the **Targets** menu, select **Databases**.
3. Select **Search List**, then click the name of a database target for which you want to create an Oracle Data Redaction policy.
4. On the home page of the database target, from the **Security** menu, select **Data Redaction**.
5. If you are prompted to do so, then log in to the database as a user who has the `EXECUTE` privilege on the `DBMS_REDACT` PL/SQL package and the `ADMINISTER REDACTION POLICY` system or schema privilege.
6. On the Oracle Data Redaction page, select the **Expressions** tab.
7. Select the policy expression that you want to delete, and then click **Delete**.

   The Delete Expressions confirmation dialog box appears.

8. Click **OK**.

Glossary

Index