Sagar Giri, 0205 | Section B

Virtual Private Database And policy types

Virtual Private Database (VPD) is a feature of Oracle Database 11g Enterprise Edition, and was introduced in Oracle8i. It is one of the most popular security features in the database. A virtual private database or VPD masks data in a larger database so that only a subset of the data appear to exist, without actually segregating data into different tables, schema or databases. Oracle Virtual Private Database enables us to create security policies to control database access at the row and column level.

Essentially, Oracle Virtual Private Database adds a dynamic WHERE clause to a SQL statement that is issued against the table, view. VPD policies can be simple or complex depending on the security requirements. A simple VPD example might restrict access to data during business hours and a more complex VPD example might read an application context during a login trigger and enforce row level security against the table. No matter how users connect to the protected table (via an application, a Web interface or SQL*Plus), the result is the same.

Example:
A customer can only see his orders in the 'orders' table (below), when he is listed in the 'customers' table (above)Column Relevance

CUST_FIRST_NAME	CUST_LAST_NAME	CUSTOMER_ID
Matthias	Hannah	106

ORDER_DATE	CUSTOMER_ID	ORDER_TOTAL
31-AUG-99 09.19.37.811132 AM	105	22150.1
20-MAR-96 05.18.21.862632 PM	106	5546.6
01-AUG-00 10.22.48.734526 AM	106	2075.2
31-AUG-99 08.53.06.008765 PM	107	70576.9

Column Relevance

With "Column Relevance", VPD can be configured such that the policy is enforced only when a critical column is selected. For example: The account manager with the account_mgr_id "149" can see all rows from the customers table, but not the credit limits. As soon as she queries the 'credit_limit' column, she can only see her own customers.

CUST_LAST_NAME	CUST_FIRST_NAME	ACCOUNT_MGR_ID
Roberts	Ishwarya	145
Steenburgen	Gustav	145
Olin	Hal	147
Kanth	Hannah	147
Seignier	Blake	149
Powell	Claude	149

CUST_LAST_NAME	CUST_FIRST_NAME	CREDIT_LIMIT	ACCOUNT_MGR_ID
Seignier	Blake	1200	149
Powell	Claude	1200	149

Column Hiding

The most advanced configuration "Column Hiding" of VPD allows for the most effective combination of ease-of-use and security: User still has access to all public information in the 'customers' table, but confidential information remains hidden.

Working mechanism of VPD:

- When a user directly or indirectly accesses a table, view, or synonym that is protected with an Oracle Virtual Private Database policy, Oracle Database dynamically modifies the SQL statement of the user.
- This modification creates a dynamic WHERE condition (called a predicate) returned by a function implementing the security policy.
- Oracle Database modifies the statement dynamically, transparently to the user, using any condition that can be expressed in or returned by a function.

Virtual Private Database policies

We can apply Oracle Virtual Private Database policies to SELECT, INSERT, UPDATE, INDEX, and DELETE statements.
For example, suppose a user performs the following query:

```
SELECT * FROM OE.ORDERS;
```

The Oracle Virtual Private Database policy dynamically appends the statement with a WHERE clause. For example:

```
SELECT * FROM OE.ORDERS
WHERE SALES_REP_ID = 159;
```

In this example, the user can only view orders by Sales Representative 159.

We can make the query more secure if we want is to filter the user based on the session information of that user, such as the ID of the user, we can create the WHERE clause to use an application context.
For example:

```
SELECT * FROM OE.ORDERS
WHERE SALES_REP_ID = SYS_CONTEXT('USERENV','SESSION_USER');
```

Components of an Oracle Virtual Private Database Policy

To implement Oracle Virtual Private Database, we must create a function to generate the dynamic WHERE clause, and a policy to attach this function to the objects that you want to protect. To generate the dynamic WHERE clause (predicate), you must create a function (not a procedure) that defines the restrictions that you want to enforce. Usually, the security administrator creates this function in his or her own schema.

Sagar Giri, 0205 | Section B

Behavior for a function

- It must take as arguments a schema name and an object (table, view, or synonym) name as inputs.
- It must provide a return value for the WHERE clause predicate that will be generated. The return value for the WHERE clause is always a VARCHAR2 data type.
- It must generate a valid WHERE clause.
- It must not select from a table within the associated policy function. Although we can define a policy against a table, we cannot select that table from within the policy that was defined against the table.

There are Five Oracle Virtual Private Database Policy Types.

DYNAMIC : Policy function re-executes every time a policy-protected database object is accessed.

STATIC : Once, then the predicate is cached in the SGA (System Global Area)

SHARED_STATIC: Same as STATIC

CONTEXT_SENSITIVE : At statement execution time when the local application context changed since the last use of the cursor

SHARED_CONTEXT_SENSITIVE : Same as CONTEXT_SENSITIVE, but multiple objects can share the policy function from the session UGA (User Global Area).

Creating Oracle Virtual Private Database Policies

Step 1:
Ensure That the OE User Account Is Active. Log on to SQL*Plus as user SYS with the SYSDBA privilege. Run the following SELECT statement on the DBA_USERS data dictionary view:

```
SELECT USERNAME, ACCOUNT_STATUS FROM DBA_USERS WHERE USERNAME ='OE';
```

If the DBA_USERS view lists user OE as locked and expired, then enter the following statement to unlock the OE account and create a new password:

```
ALTER USER OE ACCOUNT UNLOCK IDENTIFIED BY <password>;
```

Step 2:
Create the following Policy Function, which will append the WHERE SALES_REP_ID = 159 clause to any SELECT statement on the OE.ORDERS table.Creating Oracle Virtual Private

```
CREATE OR REPLACE FUNCTION auth_orders(
  schema_var IN VARCHAR2,
  table_var  IN VARCHAR2
 )
 RETURN VARCHAR2
 IS
  return_val VARCHAR2 (400);
 BEGIN
  return_val := 'SALES_REP_ID = 159';
  RETURN return_val;
 END auth_orders;
/
```

Step 3:
create the following policy by using the ADD_POLICY procedure in the DBMS_RLS package

```
BEGIN
  DBMS_RLS.ADD_POLICY (
    object_schema    => 'oe',
    object_name      => 'orders',
    policy_name      => 'orders_policy',
    function_schema  => 'sys',
    policy_function  => 'auth_orders',
    statement_types  => 'select, insert, update, delete'
   );
 END;
/
```

Now verify the policy has been implemented or not. Log on as user OE.

```
> CONNECT oe
> Enter password: password
```

Executing select/ count command will not return all the entries.

Remove the Components As user SYS, remove the function and policy as follows:

```
DROP FUNCTION auth_orders;
EXEC DBMS_RLS.DROP_POLICY('OE','ORDERS','ORDERS_POLICY');
```

Sources:
https://docs.oracle.com/cd/B28359_01/network.111/b28531/vpd.htm#CIHIFFDE
http://web.stanford.edu/dept/itss/docs/oracle/10g/network.101/b10773/apdvpoli.htm
http://www.devshed.com/c/a/oracle/row-level-security-with-virtual-private-database