Database Security and Auditing: Protecting Data Integrity and Accessibility

Chapter 6
Virtual Private Databases

Objectives

- Define the term "virtual private database" and explain its importance
- Implement a virtual private database by using the VIEW database object
- Implement a virtual private database by using Oracle's application context
- Implement row-level and column-level security

Why VPD?

- Security
  - Server-enforced security (as opposed to application-enforced).
- Purposes/benefits:
  - Security requirements necessitate data access be restricted at row or column level (FGA)
  - One database schema serves multiple unrelated groups or entities

Why VPD?

- Scalability
  - Table **Customers** contains 1,000 customer records.
  - Suppose we want customers to access **their own records only**.
  - Using views, we need to create 1,000 views. Using VPD, it can be done with a single policy function.
- Simplicity
  - Say, we have a table **T** and many views are based on **T**.
  - Suppose we want to restrict access to some information in T.
  - Without VPD, all view definitions have to be changed.
  - Using VPD, it can be done **by attaching a policy function to T**; as the policy is enforced in T, the policy is also enforced for all the views that are based on T.

Overview of Virtual Private Databases

- A VPD deals with data access
- VPD controls data access at the row or column level
- Oracle10/11*g:*
  - **Fine-grained access control**: associate security policies to database objects
  - **Application Context**: define and access application or session attributes
  - Combining these two features, VPD enables administrators to define and enforce row-level access control policies based on session attributes.
- Implementing Row- and Cell-Level Security in Classified Databases Using SQL Server 2005 http://technet.microsoft.com/en-us/library/cc966395.aspx

Overview of Virtual Private Databases (continued)

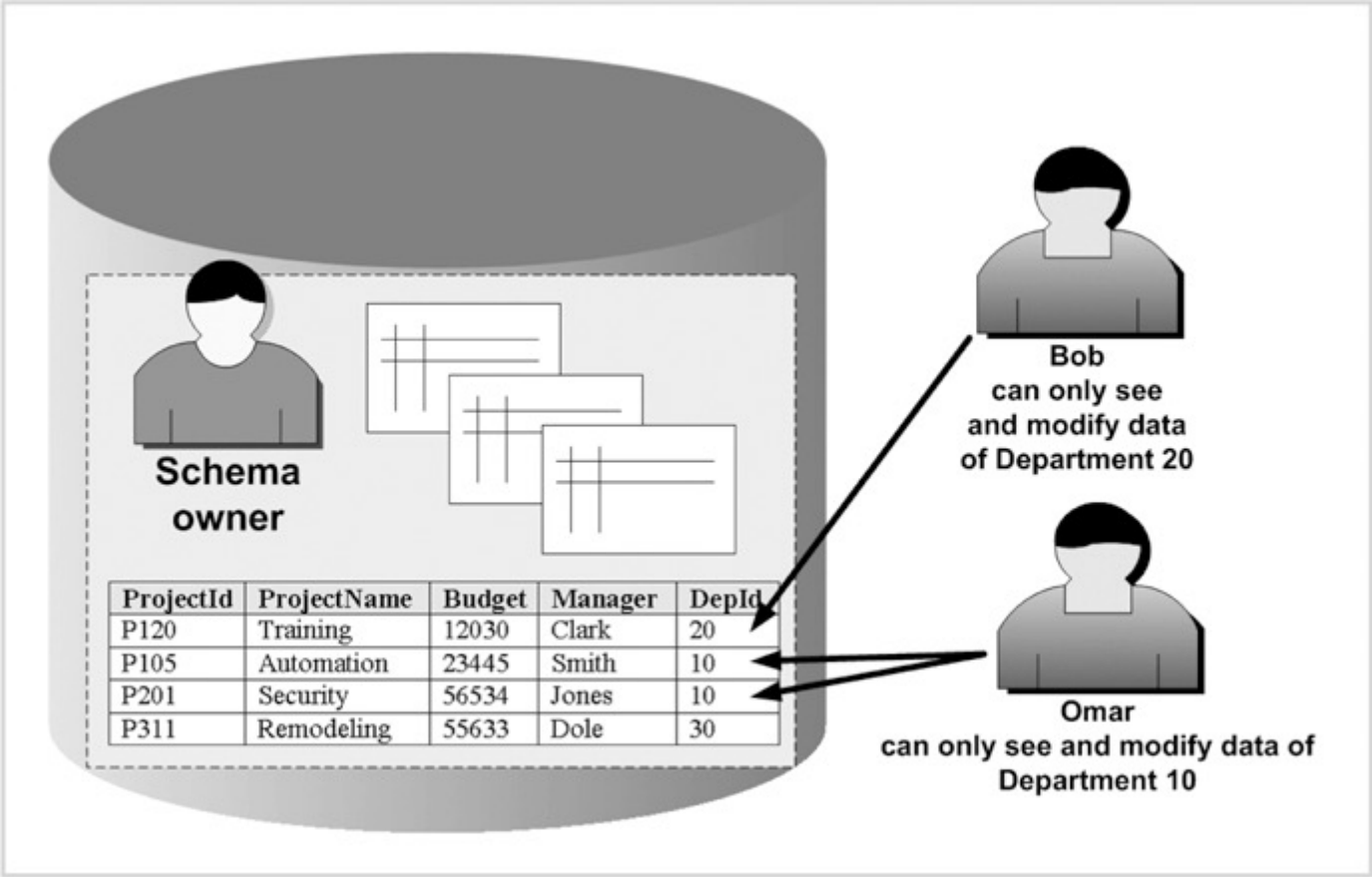


FIGURE 6-2 Virtual private database example

Implementing a VPD Using Views

Implementing a VPD Using Views

- View object limits what users can see and do with existing data: hides columns or rows from users
- **CREATE VIEW** statement: creates data views

Implementing a VPD Using Views

- Example implementation steps:
  - Logon as DBSEC schema
  - Display the EMPLOYEES table
  - Create the table EMPLOYEES_VER1

```
CREATE TABLE EMPLOYEES_VER1
(
   EMPLOYEE_ID                    NUMBER(6),
    FIRST_NAME                    VARCHAR2(20),
    LAST_NAME                     VARCHAR(2),
   EMAIL                          VARCHAR2(25),
   PHONE_NUMBER                   VARCHAR2(20),
   HIRE_DATE                      DATE,
   JOB_ID                         VARCHAR2(10),
   SALARY                         NUMBER(8, 2),
   MANAGER_ID                     NUMBER(6),
   DEPARTMENT_ID                  NUMBER(4),
   CTL_UPD_USER                   VARCHAR2(30)
)
```

Implementing a VPD Using Views

- Create a VIEW object to display rows that belong only to the logged on user

```
CREATE VIEW EMPLOYEES_VIEW1 AS
SELECT EMPLOYEE_ID, FIRST_NAME,
  LAST_NAME, EMAIL, PHONE_NUMBER,
  HIRE_DATE, JOB_ID, SALARY, MANAGER_ID,
  DEPARTMENT_ID, CTL_UPD_USER USER_NAME
FROM EMPLOYEES_VER1
WHERE CTL_UPD_USER = USER
```

Rename to USER_NAME

Implementing a VPD Using Views

- Grant SELECT and INSERT on this view to another user

```
GRANT SELECT, INSERT ON EMPLOYEE_VEIEW1 TO SCOTT
```

- Insert a row using EMPLOYEES_VIEW1

```
INSERT INTO DBSEC.EMPLOYEES_VIEW1(EMPLOYEE_ID,
  FIRST_NAME, LAST_NAME, EMAIL, PHONE_NUMBER,
  HIRE_DATE, JOB_ID, SALARY, MANAGER_ID,
  DEPARTMENT_ID, USER_NAME)
VALUES(100, 'SAM', 'AFYOUNI', 'SAFYOUNI',
  '123.234.3456', SYSDATE, 'WM_CLK', 1000, 1000,
  10, USER);
```

- USER is a function that returns the user name value of the person who is logged on.
- If log on as DESEC, USER = DBSEC
- If log on as SCOTT, USER = SCOTT

Implementing a VPD Using Views

- Example implementation steps (continued)
  - Logon as the other user
  - Select the `EMPLOYEES_VIEW1` VIEW object; you see only rows that belongs to the other user

Implementing a VPD Using Views

- Alternatively, add a trigger on insert to populate the user name automatically
- A trigger is a stored PL/SQL procedure that fires (is called) automatically when a specific event occurs, such as the `BEFORE INSERT` event.

```
CREATE OR REPLACE TRIGER
  TRG_EMPLOYEES_VER1_BEFORE_INS
BEFORE INSERT
ON EMPLOYEES_VER1
FOR EACH ROW
BEGIN
  : NEW.CTL_UPD_USER := USER;
END;
```

Implementing a VPD Using Views

Trigger is fired here to insert NEW.CTL_UPD_USER := USER

```
INSERT INTO
  DBSEC.EMPLOYEES_VIEW1(EMPLOYEE_ID,
  FIRST_NAME, LAST_NAME, EMAIL,
  PHONE_NUMBER, HIRE_DATE, JOB_ID,
  SALARY, MANAGER_ID, DEPARTMENT_ID)
VALUES(100, 'SAM', 'AFYOUNI',
  'SAFYOUNI', '123.234.3456', SYSDATE,
  'WM_CLK', 1000, 1000, 10);
```

The above statement will automatically update field CTL_UPD_USER with USER because of the trigger **TRG_EMPLOYEES_VER1_BEFORE_INS**.

Implementing a VPD Using Views

- Views can become hard to administer; solution is VPD
- Implementation is limited and requires careful design and development

Oracle VPD

Oracle VPD

- How does it work?

  When a user accesses a table (or view or synonym) which is protected by a VPD policy (function),

  1. The Oracle server invokes the policy function whenever a logged on user tries to execute a query.
  2. The policy function returns a predicate, based on session attributes or database contents.
  3. The server dynamically rewrites the submitted query <u>by appending the returned predicate to the WHERE clause.</u>
  4. The modified SQL query is executed.

Oracle VPD: Example

- Suppose Alice has the following table.

  my_table(owner varchar2(30), data varchar2(30));

- Users can access only the data of their own. But Admin should be able to access any data without restrictions.

Oracle VPD: Example

1. Create a policy function

```
Create function sec_function(p_schema varchar2, p_obj varchar2)
Return varchar2
As
    user VARCHAR2(100);
Begin
    if ( SYS_CONTEXT('userenv', 'ISDBA') ) then
            return ' ';          //Admin can access any data
    else
            user := SYS_CONTEXT('userenv', 'SESSION_USER');
            return 'owner = ' || user;
            // Users can only access their own data
    end if;
End;
```

// userenv = the pre-defined application context

Oracle VPD: Example

2. Attach the policy function to my_table

```
execute dbms_rls.add_policy (object_schema => 'Alice',
                             object_name => 'my_table',
                             policy_name => 'my_policy',
                             function_schema => 'Alice',
                             policy_function => 'sec_function',
                             statement_types => 'select',
                                                update, insert',
                             update_check => TRUE );
```

Oracle VPD-Example

3. Bob accesses my_table

select * from my_table;
=> select * from my_table where owner = 'bob';
: only shows the rows that owner is 'bob'

insert into my_table values('bob', 'Some data'); OK!

insert into my_table values('alice', 'Other data'); NOT OK!
= because of the check option.

Column-level VPD

- Instead of attaching a policy to a whole table or a view, attach a policy only to security-relevant columns
  - Default behavior: restricts the number of rows returned by a query.
  - Masking behavior: returns all rows, but returns NULL values for the columns that contain sensitive information.

- Restrictions
  - Applies only to 'select' statements
  - The predicate must be a simple boolean expression.

Column-level VPD: Example

- Suppose Alice has the following table.

  Employees(e_id number(2), name varchar2(10), salary nubmer(3));

e_id	Name	Salary
1	Alice	80
2	Bob	60
3	Carl	99

- Users can access e_id's and names without any restriction. But users can access only their own salary information.

Column-level VPD: Example

1. **Create a policy function**

```
Create function sec_function(p_schema varchar2, p_obj varchar2)
Return varchar2
As
   user VARCHAR2(100);
Begin
   user := SYS_CONTEXT('userenv', 'SESSION_USER');
   return 'Name = ' || user;
   end if;
End;
```

Column-level VPD: Example

2. Attach the policy function to Employees (default behavior)

```
execute dbms_rls.add_policy (object_schema => 'Alice',
                             object_name => 'employees',
                             policy_name => 'my_policy',
                             function_schema => 'Alice',
                             policy_function => 'sec_function',
                             sec_relevant_cols=>'salary');
```

Column-level VPD: Example

3. Bob accesses table Employees (default behavior)

select e_id, name from Employee;

e_id	Name
1	Alice
2	Bob
3	Carl

select e_id, name, salary from Employee;

e_id	Name	Salary
2	Bob	60

Column-level VPD: Example

2'. Attach the policy function to Employees (masking behavior)

```
execute dbms_rls.add_policy (object_schema => 'Alice',
                             object_name => 'employees',
                             policy_name => 'my_policy',
                             function_schema => 'Alice',
                             policy_function => 'sec_function',
                             sec_relevant_cols=>'salary',
                 sec_relevant_cols_opt=>dbms_rls.ALL_ROWS);
```

Column-level VPD: Example

3. Bob accesses table Employees (masking behavior)

select e_id, name from Employee;

e_id	Name
1	Alice
2	Bob
3	Carl

select e_id, name, salary from Employee;

e_id	Name	Salary
1	Alice	
2	Bob	60
3	Carl	

Implementing a VPD Using Application Context in Oracle

Application Context

- Application contexts act as secure caches of data that may be used by a fine-grained access control policy.
  - Upon logging into the database, Oracle sets up an application context in the user's section.
  - You can define, set and access application attributes that you can use as a secure data cache.
- There is a pre-defined application context, *"userenv"*.
  - in Oracle Security Guide.

Implementing a VPD Using Application Context in Oracle (continued)

Table 6-1 Common USERENV namespaces

Attribute	Description of What the Attribute Returns
TERMINAL	Operating system terminal name for the current connected session
IP_ADDRESS	Network IP address for the current connected session
HOST	Name of the host machine for the current connected session
DB_NAME	Name of the database to which the current session is connected
CURRENT_USER	Database name for the current connected session
DB_DOMAIN	Network domain name for the database to which the current session is connected
OS_USER	Operating system user name for the current connected session
SERVER_HOST	Name of the host machine to which the current database session is connected
SESSIONID	Auditing session identifier for the current connected session
ISDBA	Information to indicate whether the connected session has DBA privileges or not; the returned value is a Boolean TRUE or FALSE

The information in this table is derived from the online documentation that Oracle provides at the Oracle Technology Network site: *www.otn.oracle.com*.

Implementing a VPD Using Application Context in Oracle

- To <u>set</u> an attribute value in an application context,

```
DBMS_SESSION.SET_CONTEXT('namespace',
'attributename', value);
```

- To <u>get</u> an attribute value from an application context,

```
SYS_CONTEXT('namespace', 'attributename');
```

Example:

```
DBMS_SESSION.SET_CONTEXT('USERENV',
  'IP_ADDRESS', "192.168.1.2");

SYS_CONTEXT('USERENV', 'IP_ADDRESS')
```

Returns 192.168.1.2

Implementing a VPD Using Application Context in Oracle

- Application context:
  - Functionality specific to Oracle
  - Allows to set database application variables that can be retrieved by database sessions
  - Variables can be used for security context-based or user-defined environmental attributes

Implementing Virtual Private Databases (continued)

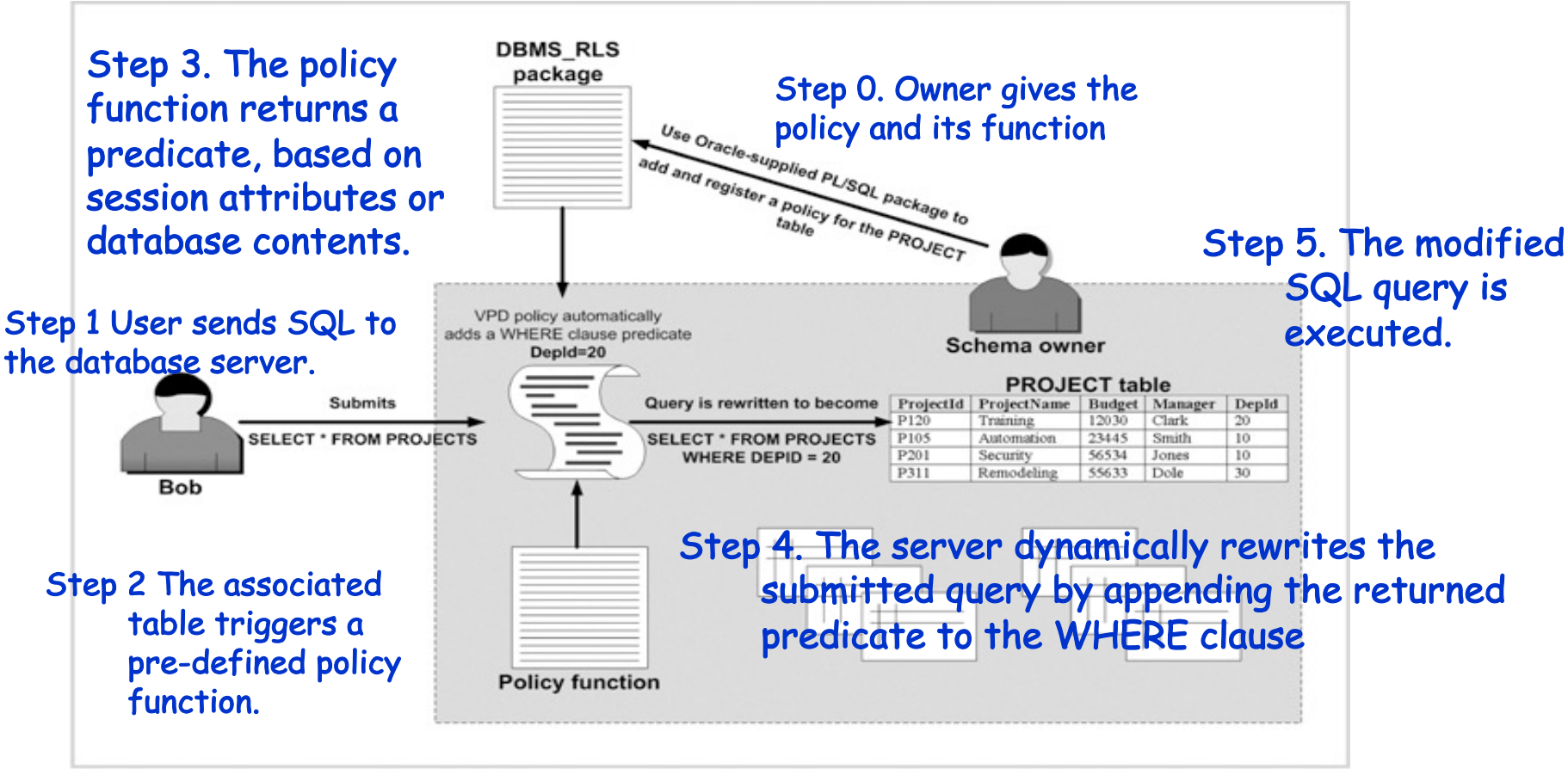
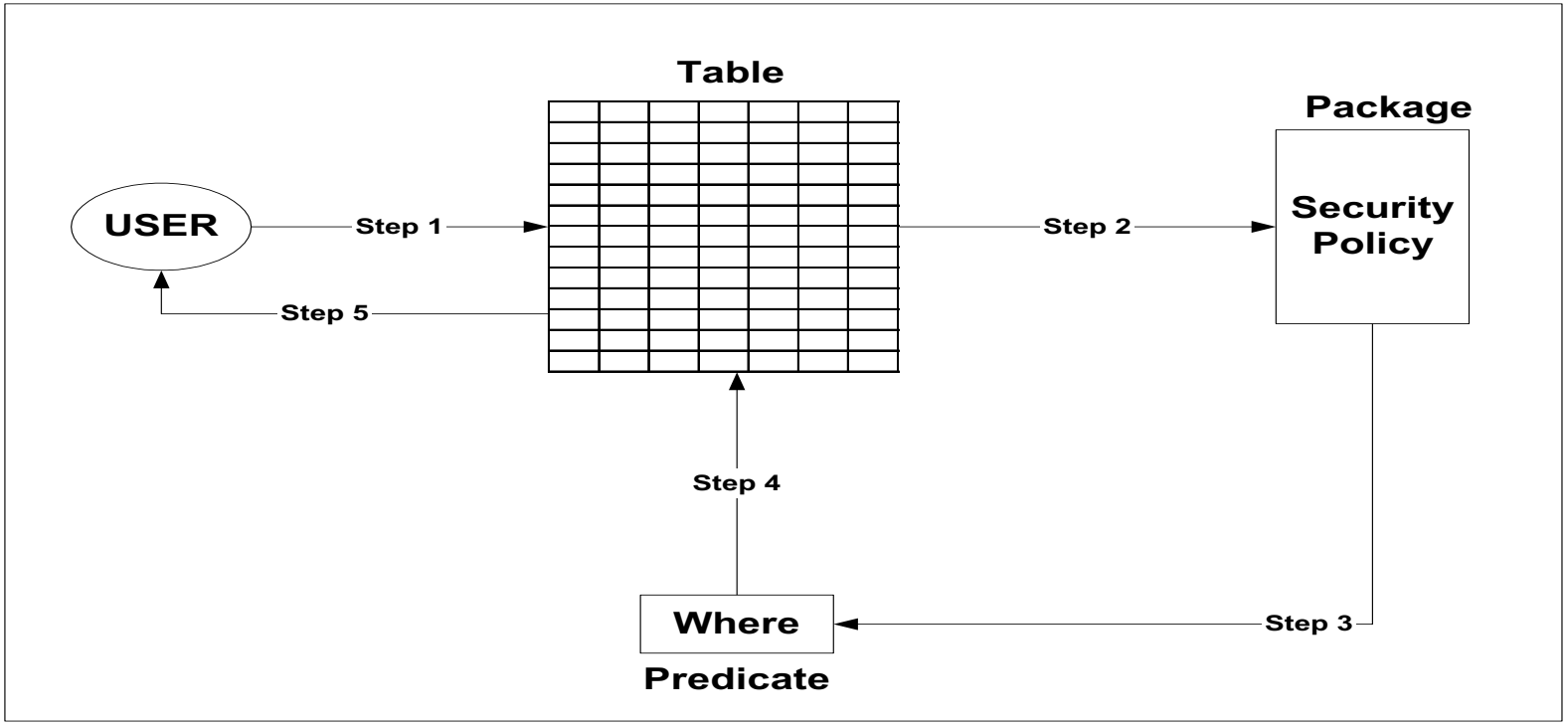


FIGURE 6-6 Architecture of Oracle virtual private database feature

Virtual Private Database Technology

Data access via Virtual Private Database will perform the following five steps:

1. User sends SQL to the database server.
2. The associated table triggers a pre-defined security policy.
3. The security policy returns a predicate.
4. The SQL statement is modified according to the security policy.
5. Secured data returns to user.



Application Context

- One can create a customized application context and attributes.
  - Say, each employee can access a portion of the Customers table, based on the job-position.
  - For example, a clerk can access only the records of the customers who lives in a region assigned to him. But a manager can access any record.
  - Suppose that the job-positions of employees are stored in a LDAP server (or in the Employee table).
  - Such information can be accessed and cached in an application context when an employee logs in.

Create Application Context

1. Create a PL/SQL package that sets the context

```
Create package Emp_env_context IS
      procedure Set_job_position IS
        jp varchar(100);
      begin
         select job_pos into jp from Employee
         where name = SYS_CONTEXT('USERENV', 'SESSION_USER');
         DBMS_SESSION.SET_CONTEXT('emp_env', 'job', jp);
      end;
End;
```

2. Create a context and associate it with the package

```
Create Context emp_env Using Emp_env_context;
```

- Any attribute in the "emp_env" context can only be set by procedures in the "Emp_env_context" package.

Using Application Context

3. Set the context before users retrieve data (at the login)

```
Create or Replace Trigger Emp_trig
    After Logon On Database
    Begin
        Emp_env_context. Set_job_position
    End
```

- Use an event trigger on login to pull session information into the context.

4. Use the context in a VPD function

```
if (SYS_CONTEXT('emp_env', 'job') = 'manager')
    return '';
else …
```

Multiple Policies

- It is possible to associate multiple policies to a database object.
  - The policies are enforced with AND syntax.
  - For example, suppose table T is associated with {P1, P2, P3}.
  - When T is accessed by query Q = select A from T where C.
  - Q' = select A from T where $C \wedge (c1 \wedge c2 \wedge c3)$.

- Different from Stonebraker's approach
  - The policies are enforced with OR syntax.
  - Q' = select A from T where $C \wedge (c1 \vee c2 \vee c3)$.

- While Stonebraker's policies specify "what users can see" (permissions), VPD policies specify "what users cannot see" (prohibitions).

Viewing VPD Policies and Applications Context Using Policy Manager

- Graphical tool called Policy Manager
- Use SYSTEM credentials to log in
- FGA control policies are divided into two parts:
  - Policy groups
  - Application context

Viewing VPD Policies and Applications Context Using Policy Manager (continued)

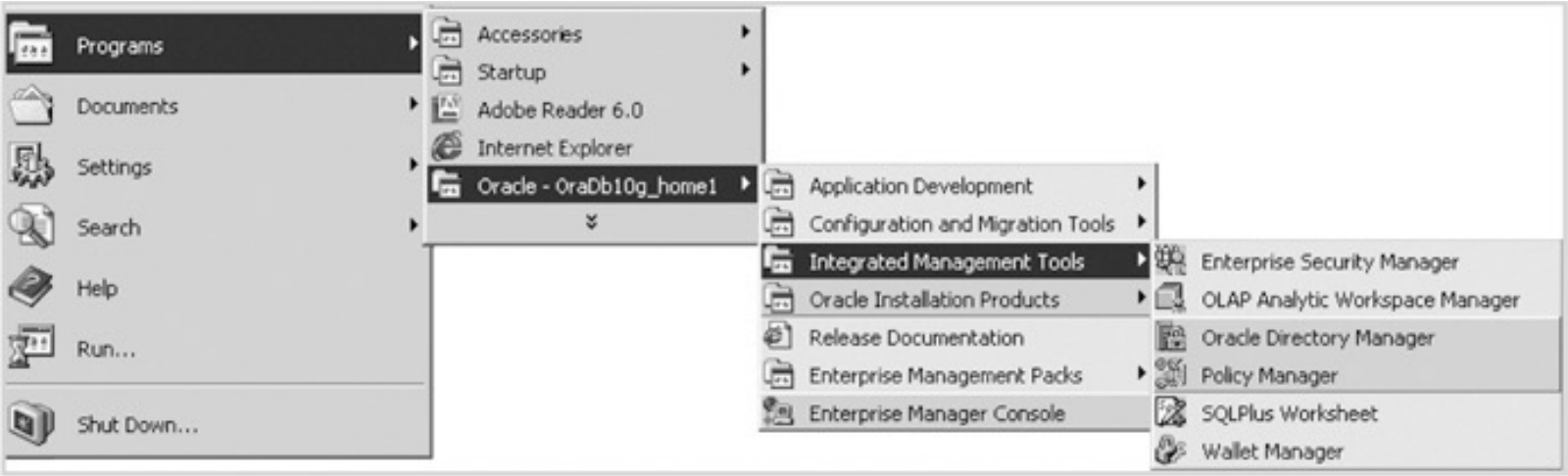


FIGURE 6-7 Policy Manager shortcut in the Start menu

Viewing VPD Policies and Applications Context Using Policy Manager (continued)

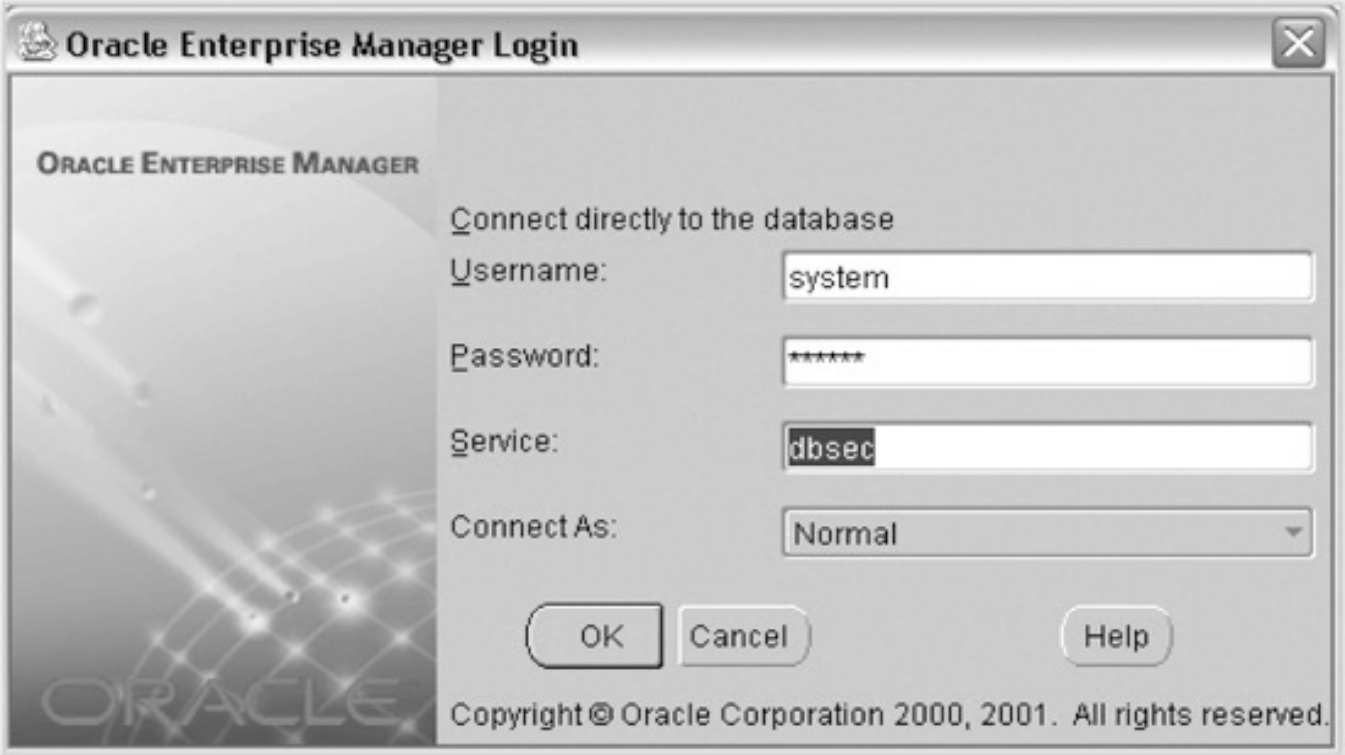


FIGURE 6-8 Logging into Oracle Policy Manager

Viewing VPD Policies and Applications Context Using Policy Manager (continued)

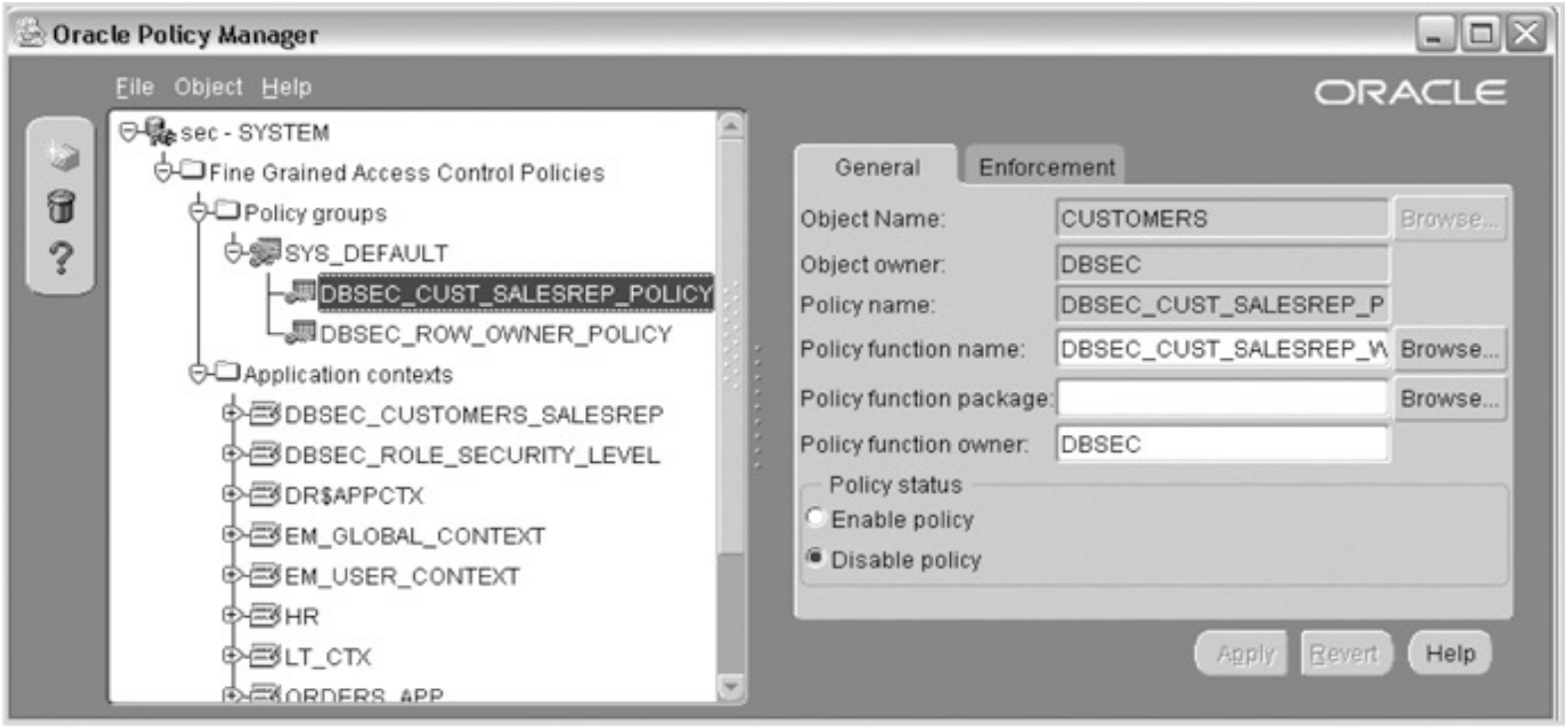


FIGURE 6-9 Oracle Policy Manager

Another Example from Textbook

DBSEC_CUSTOMER_APP_CONTEXT Table

SALES_REP_ID	USER_NAME
2336	VPD_CLERK1
9644	**VPD_CLERK2**
4587	VPD_CLERK3

DBSEC_CUSTOMERS_SALESREP Context
Buffer/cache

SELES_REP_ID is set '9644' based on SALES_REP_ID of logon user who is VPD_CLERK2. '9644' is retrieved from predefined table DBSEC_CUSTOMER_APP_CONTEXT table based on USER_NAME = 'VPD_CLERK2', when trigger is executed right after VPD_CLERK2 logs on.

Function creates WHERE clause 'SALES_REP_ID'= = 9644' where '9644' is retrieved from DBSEC_CUSTOMERS_SALESREP Context.

1. user VPD_CLERK2 logs on.

Trigger executes PKG_DBSEC_CUST_SALES_REP. SET_CONTEXT

2. User send query SELECT * FROM CUSTOMERS

Security policy attached to table CUSTOMERS is checked. **Function** of security policy is consulted.

3. Query is rewritten: SELECT * FROM CUSTOMERS WHERE SALES_REP_ID=9644

Page 232-234

Summary

- A virtual private database allows or prevents data access at the row or column level; implemented using VIEW database object
- VPDs are also referred to as row-level security (RLS) or fine-grained access (FGA)
- SQL Server does not support VPDs
- Microsoft SQL Server 2000 system function of USER

Summary (continued)

- Oracle Application context:
  - Allows setting of database application be retrieved by database sessions
  - SYS_CONTEXT function
  - PL/SQL package DBMS_SESSION
  - SET_CONTEXT procedure
- Use Oracle-supplied package DBMS_RLS to add the VPD policy
- Oracle data dictionary views

Summary (continued)

- Oracle Policy Manager: graphical tool used to administer VPD policies
- Oracle has the capability to restrict updates or inserts on columns, using GRANT UPDATE(column) and INSERT(column)

Resources and Lab3

- Animated DataBase Courseware

http://adbc.kennesaw.edu/

- Oracle VPD

http://www.oracle.com/technetwork/database/security/ols-sar1-084290.html

- Lab 3:

http://www.oracle.com/technetwork/articles/idm/vpd-otn-099555.html